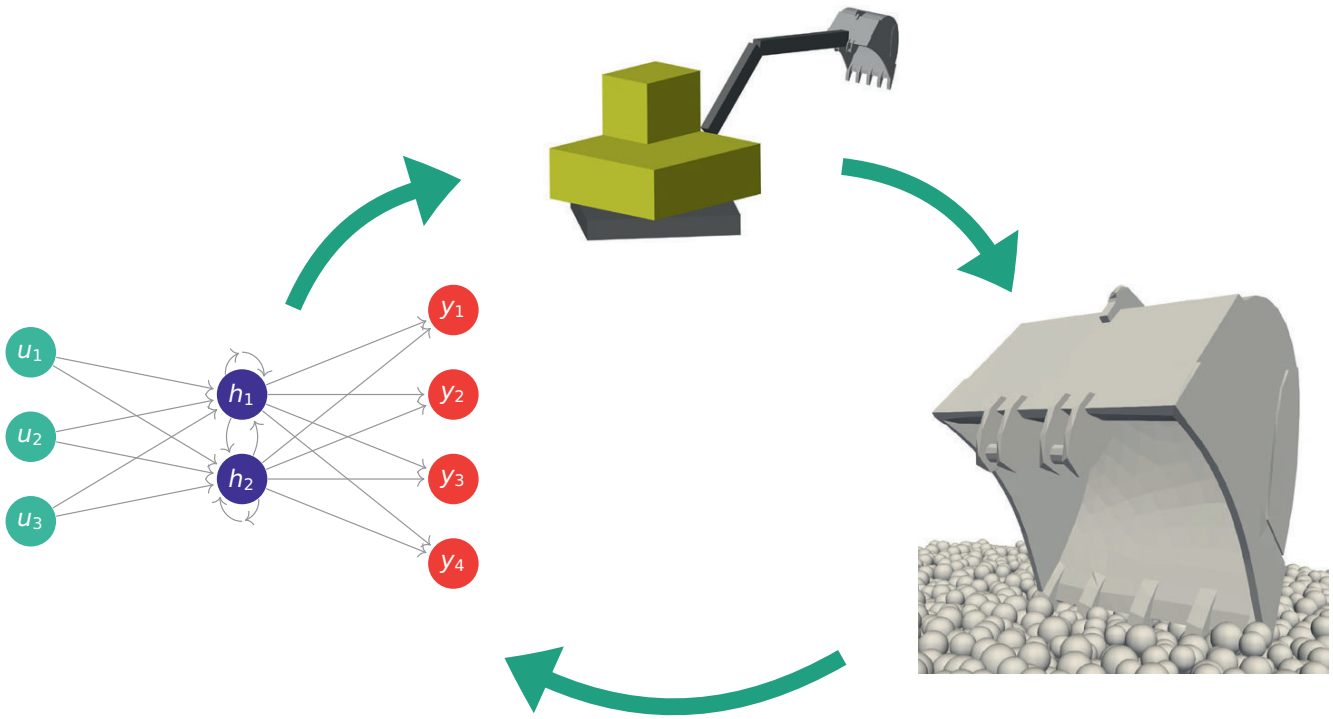
Fraunhofer
ITWM

Fraunhofer Institute for Industrial
Mathematics ITWM



Efficient Numerical Simulation of Soil-Tool Interaction

Jonathan Jahnke

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Fraunhofer Verlag

Fraunhofer Institute for
Industrial Mathematics ITWM

Efficient Numerical Simulation of Soil-Tool Interaction

Jonathan Jahnke

Fraunhofer Verlag

Contact:
Fraunhofer Institute for
Industrial Mathematics ITWM
Fraunhofer-Platz 1
67663 Kaiserslautern
Germany
Phone +49 631/31600-0
info@itwm.fraunhofer.de
www.itwm.fraunhofer.de

Bibliographic information of the German National Library:
The German National Library has listed this publication in its Deutsche Nationalbibliografie; detailed bibliographic data is available on the internet at www.dnb.de.

ISBN 978-3-8396-1835-6

DE-386

Zugl.: Kaiserslautern, TU, Diss., 2022

Print and finishing:
Fraunhofer Verlag, Mediendienstleistungen

The book was printed with chlorine- and acid-free paper.

Nobelstrasse 12
70569 Stuttgart
Germany
verlag@fraunhofer.de
www.verlag.fraunhofer.de

is a constituent entity of the Fraunhofer-Gesellschaft, and as such has no separate legal status.

Fraunhofer-Gesellschaft zur Förderung
der angewandten Forschung e.V.
Hansastrasse 27 c
80686 München
Germany
www.fraunhofer.de

Efficient Numerical Simulation of Soil-Tool Interaction

Vom Fachbereich Mathematik der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

von
Jonathan Jahnke

Gutachter: **Prof. Dr. Bernd Simeon**
Prof. Dr.-Ing. Prof. E.h. Peter Eberhard

Datum der Disputation: 18. März 2022

DE-386

Dedicated in gratitude to my grandparents,
who ignited the spark of science.

Acknowledgement

First, I want to thank Bernd Simeon for accepting me as a doctoral student and for the continuous supervision in the PhD-Seminar. Second, I am grateful to Klaus Dreßler and Michael Burger, heads of the department *Dynamics, Loads and Environmental Data* of the division *Mathematics for Vehicle Engineering* at *Fraunhofer Institute for Industrial Mathematics ITWM* for the financial support in the last years. I especially thank Stefan Steidel and Michael Burger for the supervison of my work, especially during our regular meetings in the last year.

Stefan always had an open ear if there was anything to discuss and highly motivated me in difficult times. Michael kindly introduced me into the world of machine learning. Thanks go to Steffen for the fruitful discussions on particle models and other matters. Thereafter, I want to thank all of my colleagues at ITWM, and especially MF for the nice and friendly atmosphere, which sadly made it even more difficult in times of the pandemic not to come to the office and work from home. I highly appreciated the planned and spontaneous coffee breaks and daily lunch with the other colleagues and PhD students from ITWM. Particularly, I want to thank Michael Kleer, Christoph Mühlbach, Andrey Gizatullin and especially Sebastian Emmerich and René Reinhard from the RODOS developer team. Michael and Christoph supported me with the setup of the bucket experiment for the soil laboratory and had always good advice on technical problems. Thanks go to Ronald and Maurice from the soil laboratory and also to Christos Vrettos and Andreas Becker.

I wish to thank my predecessors in the particle simulation group at ITWM, namely Matthias Balzer, Jan Kleinert, Martin Obermayr, and Dietmar Weber, who developed the routines and the software, forming the basis of this work. The time at DLR with Jan Kleinert in 2019 was fun and instructive.

Thanks to Ben, Jan, Matthias, Sebastian, Simon, Steffen, Torsten and Urs for proofreading parts or early versions of this thesis and for countless advice on LaTeX and on the art of academic writing.

I am grateful to my family and friends for supporting me in all these years and for always having an open ear. Last but not least, I want to thank my wife, Isabell, for her continuous support and love.

Abstract

Force predictions of soil-tool interaction using the Discrete Element Method (DEM) are widely established. In addition to an acceptable prediction quality, the efficient simulation of granular material on high performance clusters with modern parallelization strategies for the industrial application is indispensable. But for relevant problem sizes such simulations are so far not realtime capable. Further on, the inclusion of the human-machine interaction at a driving simulator combined with soil-tool simulation poses many interesting research questions. We therefore strive for sufficient performance regarding computation time. Consequently, we consider alternative models and algorithms to achieve realtime capability.

In this thesis, we focus on the interaction of a digging tool with soil, and consider the example of an excavator bucket interacting with a soil trench. We therefore study a variety of modeling approaches to enable accurate soil-reaction force predictions in realtime. First, we compare three classes of particle contact models. We examine not only the suitability for accurate realtime prediction, but also evaluate the particle models with respect to accurate and efficient generation of training data for machine learning models. The classical penalty DEM solves Newton's law on an acceleration level. Nonsmooth Contact Dynamics (NSCD) solve the equations of motion on the velocity level, require less smoothness, and allow for larger time steps. Position Based Dynamics (PBD) originates in the computer graphics community and solves the contact problem on the position level. While PBD and NSCD are more efficient, force computations are more reliable using penalty-based DEM. We further present several basic ideas regarding hybrid models.

Thereafter, we discuss two data-based methods of the field of supervised learning which allow force predictions in realtime. First, we present a DEM Lookup Table approach, which accesses previously computed data efficiently in an online phase. Second, we consider a DEM Recurrent Neural Network approach to emulate the dynamics arising in the excavation and to enhance the prediction quality for more complex excavation maneuvers. Finally, we deploy the application of the derived methods at the driving simulator RODOS and evaluate the developed algorithms regarding accuracy and practicability.

Keywords: Discrete Element Method, Nonsmooth Contact Dynamics, Position Based Dynamics, Multibody Dynamics, Data-Based Modelling, k-Nearest Neighbor Search, Recurrent Neural Networks, Machine Learning

Zusammenfassung

Zur Kraftvorhersage bei Boden-Werkzeug-Wechselwirkungen ist die Diskrete Elemente Methode (DEM) ein weit verbreiteter Ansatz. Neben akzeptabler Vorhersagegenauigkeit ist die effiziente Simulation granularer Materialien mit Hochleistungsrechnern und Parallelisierungsstrategien für die industrielle Anwendung unabdinglich. Jedoch sind solche Simulationen für relevante Problemgrößen noch nicht echtzeitfähig. Außerdem ist die Bodensimulation für die Mensch-Maschine-Interaktion an einem Fahrsimulator ein interessanter Forschungsgegenstand. Dafür ist eine hinreichende Effizienz notwendig. Aus diesem Grund betrachten wir alternative Modelle und Algorithmen, um echtzeitfähige Boden-Werkzeug-Interaktion zu ermöglichen.

In dieser Arbeit erforschen wir die Boden-Werkzeug-Wechselwirkung am Beispiel einer Baggerschaufel, die in einem Bodenbett gräbt. Hierzu untersuchen wir unterschiedliche Modellierungsansätze, um mögliche echtzeitfähige Lösungsstrategien mit hoher Genauigkeit zu entwickeln. Zunächst vergleichen wir drei Algorithmenklassen, um den Partikelkontakt aufzulösen. Dazu überprüfen wir die Partikelmodelle hinsichtlich ihrer Eignung zur Kraftvorhersage in Echtzeit, und bewerten diese in Bezug auf effiziente Generierung verlässlicher Trainingsdaten für Modelle des maschinellen Lernens. Die klassische DEM löst Newtons Gesetz auf Beschleunigungsebene. Nichtglatte Methoden (NSCD) formulieren den Partikelkontakt auf Impuls-Geschwindigkeitsebene, erfordern geringere Glattheit und erlauben größere Zeitschrittweiten. Positionsbasierte Dynamik (PBD) ist eine Simulationsmethode aus dem Gebiet der Computergrafik und löst den Kontakt auf Positionsebene auf. Obwohl PBD und NSCD effizienter sind, ist die klassische DEM für Kraftberechnungen besser geeignet. Weiter diskutieren wir verschiedene, grundlegende Ideen bezüglich hybrider Modelle.

Daraufhin präsentieren wir zwei datenbasierte Methoden aus dem Bereich des überwachten Lernens, die eine Prädiktion der auf das Werkzeug wirkenden Bodenkräfte in Echtzeit ermöglichen. Zum einen stellen wir einen DEM-Lookup-Tabellen-Ansatz vor, bei dem zuvor berechnete Daten, in effizienter Weise adressiert werden. Andererseits untersuchen wir einen DEM-Rekurrente-Neuronale-Netze-Ansatz, um die Dynamik des Grabvorgangs abzubilden und die Genauigkeit für komplexere Manöver zu erhöhen. Schlussendlich realisieren wir die Anwendung der hergeleiteten Methoden am Fahrsimulator RODOS und untersuchen die entwickelten Algorithmen hinsichtlich ihrer Genauigkeit und Anwendbarkeit.

Stichworte: Diskrete Elemente Methode, Nichtglatte Kontaktdynamik, Positionsbasierte Dynamik, Mehrkörperdynamik, Datenbasierte Modellierung, Lookup Tabellen, k-nächste Nachbar Suche, Rekurrente Neuronale Netze, Maschinelles Lernen

Notation

Ω	domain
$\boldsymbol{q}$	generalized coordinates, function variable
$\boldsymbol{x}$	position vector
$\boldsymbol{v}$	velocity vector
$\boldsymbol{p}$	unit quaternion, parameter vector
$\boldsymbol{g}$	constraint
δ	distance overlap
$\boldsymbol{G} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}}$	constraint derivative
$\mathcal{A}$	algebra
$C(X, Y)$	space of continuous functions from X to Y
$C^m(X, Y)$	space of m-times continuously differentiable functions from X to Y
ε^a_{∞}	absolute maximum error function
$\varepsilon^a_{\mathcal{L}^2}$	absolute $\mathcal{L}^2$-error function
$\varepsilon^a_{\mathcal{L}^1}$	absolute $\mathcal{L}^1$-error function
ε^r_{∞}	relative maximum error function
$\varepsilon^r_{\mathcal{L}^2}$	relative $\mathcal{L}^2$-error function
$\varepsilon^r_{\mathcal{L}^1}$	relative $\mathcal{L}^1$-error function
$\boldsymbol{F}, F, FX, FY, FZ$	force vector, force scalars
$\boldsymbol{T}, T, TX, TY, TZ$	torque vector, torque scalars
σ_n	normal stress
τ	shear stress
$\mathcal{N}$	normal distribution
μ_n	normal expectation value
σ	normal standard deviation

Physical Parameters

r	radius
μ	local friction coefficient
ϕ	angle of friction
ρ	density
n	porosity
E	Young modulus
d	cutting depth (LUT-parameter)
θ	angle of incidence (LUT-parameter)
v	longitudinal velocity (LUT-parameter)

List of Acronyms

DAE Differential Algebraic Equation . . . 23
DEM Discrete Element Method . . . 3
DOF Degrees of Freedom . . . 27
FNN Feedforward Neural Network . . . 17
FEE Fundamental Earthmoving Formula . . . 57
HiL Human-in-the-Loop . . . 2
IVP Initial Value Problem . . . 11
LUT Lookup Table . . . 4
MOR Model Order Reduction . . . 1
MBS Multibody System . . . 23
NSCD Nonsmooth Contact Dynamics . . . 29
ODE Ordinary Differential Equation . . . 7
PBD Position Based Dynamics . . . 29
PGJ Projected Gauss-Jacobi . . . 36
PGS Projected Gauss-Seidel . . . 46
PSD Power Spectral Density . . . 82
RNN Recurrent Neural Network . . . 4
RODOS RObot based Driving and Operation Simulator . . . 2
SDE Stochastic Differential Equation . . . 12
TPWL Trajectory Piecewise Linear . . . 16
TT Triaxial Compression Test . . . 52
GRAPE GRAnular Physics Engine . . . 5

Contents

1 Introduction

Combine particle models and data-based modeling for realtime force predictions.
One-line summary of this thesis.

The efficient simulation of physical phenomena presents a challenging task. Many engineering problems in the past century have been overcome and understood using complex modeling techniques, e.g. based on ordinary and partial differential equations or different formulations of contact. However, not only the complexity of a system is of interest, but also how to efficiently perform a simulation. The execution time of a complex computation is often limiting factor, increasing the need for numerical efficiency. The bottleneck is mostly not the imagination and creativity of researchers, but often the computational resources of computing architectures. As the construction of high-performance systems became more and more refined and the density of transistors doubled around every two years (Moore's law), it was possible to simulate large complex systems not only in two but also in three dimensional space. In recent years, this trend has slowed down and parallelization and memory access become more and more important for computation speedup. Even today, solving such large systems still requires huge computation times.

Today, industrial applications ask for realtime capable modeling techniques. In this work, we approach this problem from two different perspectives. On the one hand, we can work with existing physical models, exchange expensive features for computationally cheaper but also less accurate ones and use Model Order Reduction (MOR) to obtain realtime capability. With simplified physical models, one might run into examples, where the accuracy is low. On the other hand, the collection of data with an expensive offline model and the online application of data-based modeling approaches often yield more promising and refined results.

The latter approach has become popular in the new millennium, as machine learning techniques gained a still ongoing boost in popularity and the number of research papers in this area grew substantially. In contrast, it is often criticized that machine learning models lack a physical interpretation and a system may fail completely with a so called adversarial attack. To know the capabilities, but also the limits of a model, is an important aspect to keep in mind.

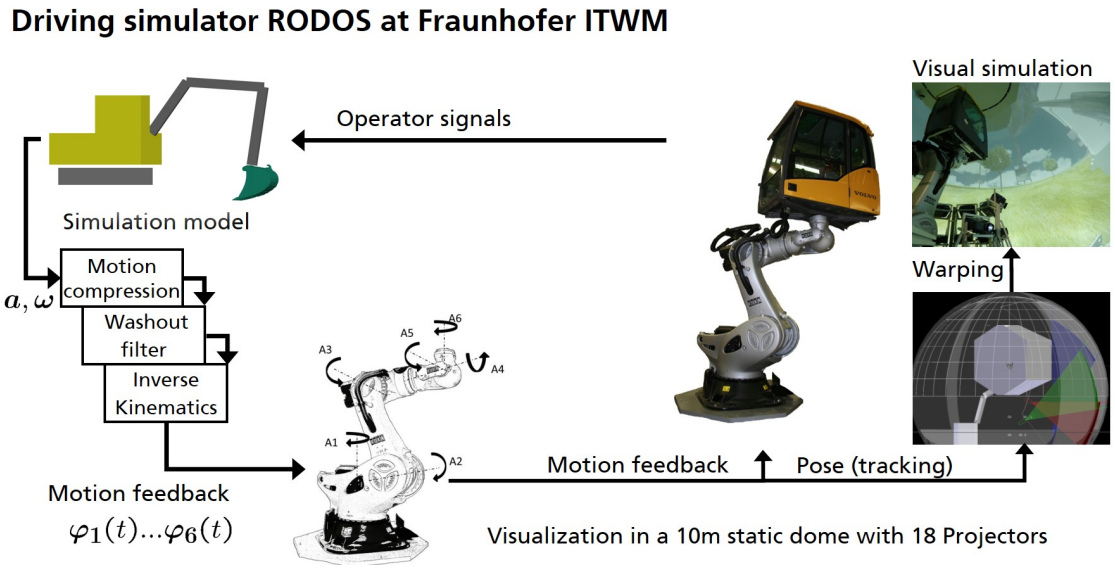


Figure 1.1: Driving Simulator RODOS at Fraunhofer ITWM, see [Kle]

Realtime capable algorithms are in demand when it comes to Human-in-the-Loop (HiL) applications. Here, an operator interacts with a machine, e.g. with a driving simulator and responds to its feedback. The RObot based Driving and Operation Simulator (RODOS) at Fraunhofer ITWM is suitable for excavation, see Figure 1.1. An operator interacts with a machine, generating a typical input signal. The operator signals are transferred to a simulation model, which computes the dynamics and motion of the vehicle. After filtering the signal, we compute the inverse kinematics, resulting in the machine actuation. The operator receives visual and motion feedback. The gap between expected motion, fitting the visualization and the experienced motion, which inevitably arises due to the limited configuration space of the machine, may cause motion sickness. The driving simulator serves several purposes. First, the simulator environment allows the reproduction of dangerous scenarios, which should be avoided in real experiments. Second, a driving assistant system can be modeled and tested in the simulator with different operators without the necessity of building expensive prototypes. Third, the development of autonomous vehicles and the drivers experience with them is another interesting application. Fourth, the driving simulator serves as an operator training site, which is less expensive compared to a full scale excavator. Also, in terms of reducing CO_2-emission, a driving simulator running on renewable electric energy is beneficial. So far, CO_2-neutral alternative engines, e.g. based on hydrogen fuel cells or fully electric, are commonly not available for construction machinery and form part of current research and exist only for prototypes, see [JCB].

We aim at realtime capable soil-tool prediction, that is we require the execution time of a simulation to be strictly smaller than the simulated time. That corresponds to a realtime factor below one, see Section 2.4.3 for a more formal definition. With our algorithms to predict correct draft forces in realtime, we pave the way for inexpensive operator training for excavation. Furthermore, the correct assessment of soil-tool forces enhances the physics of important application scenarios, e.g. excavation of a soil pit, straiten the slope of soil. This significantly improves the model fidelity in the virtual development process of earthmoving machinery, in particular of excavators.

Within this thesis, we investigate the trade-off between the computation of highly accurate reaction forces and realtime capability. Within the close future, the computer hardware impedes the computation of complex models involving thousand or even millions of particles for the prediction of forces in realtime. That is why we discuss alternative mathematical approaches to overcome the problem of accurate force predictions in realtime.

1.1 State of the Art

In this Section, we briefly review the development of particulate simulation and discuss a selection of software products incorporating particle methods for the simulation of granular material. Particle related soil simulation software has been developed since the publication of the seminal paper by Cundall and Strack [CS79] introducing the Discrete Element Method (DEM). They started by simulating disks of different radii in the 1970s and extended their approach to three dimensions in the late 1980s. In the last three decades there have been numerous advances in the field, namely the development of numerous particle models, the modeling of particles as polydisperse spheres, multi-spheres, ellipsoids, or polyhedrons, the simulation of complex application scenarios and industrial processes, just to name a few.

Soil simulation in realtime has been addressed in previous work. There have been several attempts and ideas, however most approaches fail on the accurate prediction of draft forces. Renouf et al. used nonsmooth particle methods in realtime in [RAD05], but back then succeeded only with very limited particle samples of about 100 particles. Holz et al. suggested a voxel-based approach in [HBK09]. They later enhanced their model to be applied in a driving simulator, see [HAT15]. Servin et al. proposed an adaptive merged particle approach in [SW16], a data-based approach based on regression of velocity fields in [WS21] and a terrain server with a continuum model in [SBN21].

1.2 Contributions of this Thesis

We analyze different particle models on their suitability for efficient soil-tool interaction simulation. Hybrid physical ideas are discussed as well. We propose two approaches from the field of supervised machine-learning which are realtime capable. We implement and parametrize them for the application at the RODOS, see [Kle15a].

During the progress of this work, we contributed two conference publications on a DEM Lookup Table (LUT) approach:

- Soil modeling with a DEM Lookup approach. J. Jahnke, S. Steidel and M. Burger, PAMM 2019 [JSB19]
- Efficient Particle Simulation Using a Two-Phase DEM-Lookup Approach. J. Jahnke, S. Steidel, M. Burger and B. Simeon, ECCOMAS MBD 2019 [Jah+20]

We furthermore addressed two parametrization alternatives in the following conference proceedings:

- Parameter Identification for Soil Simulation based on the Discrete Element Method and Application to Small Scale Shallow Penetration Tests. J.Jahnke, S. Steidel and M. Burger, S. Papamichael, A. Becker and C. Vrettos, PARTICLES 2019 [Jah+19]
- Triaxial Compression Tests and Direct Shear Tests in the Parametrization Procedure of Soil modeled with the Discrete Element Method. S. Steidel, J. Jahnke, X. Chang, A. Becker and C. Vrettos, PARTICLES 2021 [Ste+21]

Parts of these publications are included in Chapter 3, Chapter 4 and Chapter 5.

We evaluated three particle simulation approaches regarding their suitability for efficient soil-tool reaction force prediction. We further studied Recurrent Neural Networks (RNNs) and developed a DEM RNN approach.

1.3 Scope of this Thesis

I expect you, dear reader of this thesis, to have studied at least one year of undergraduate mathematics and to be familiar with or at least keen to learn the basic concepts in numerical mathematics, soil mechanics and technical mechanics. Engineers, interested in soil mechanical applications, who have followed introductory courses on mathematics are also encouraged to read this thesis. When we use

more advanced mathematics, we try to be self-contained if possible and refer the reader to the literature. When proofs are mentioned or non-essential mathematics is required to understand, we mention the main results. Many concepts in Chapter 2 are only touched briefly, without discussing the theory in depth.

Our goal is not to cover every aspect in full detail, but to transport the main ideas on the different roads towards realtime capable soil simulation. The aim of this thesis is the description of the relevant concepts and promising alternatives for the application of realtime force prediction with special focus on the application at an excavation driving simulator.

1.4 Outline

In this thesis, we present two fundamentally different roads to a realtime capable soil-tool interaction model based upon particle-based soil models. On the one hand, we have the acceleration and simplification of existing particle models. On the other hand, we precompute relevant data offline and rely upon methods of supervised learning. In Chapter 2, we lay out some mathematical groundwork needed for the following. In Chapter 3, we present three different common algorithms to model particulate systems. We compare them in terms of stability, efficiency and robustness. In Chapter 4, we aim towards more efficient models for soil-tool interaction. Therefore, we look at a simple two dimensional model and point out its limitations. We use our insight from the previous chapter to formulate some ideas regarding hybrid particle models. Finally, we present two data-based models, namely a DEM LUT approach and a DEM RNN approach. In Chapter 5, we present further results regarding the parametrization. Further, we investigate a bucket trench experiments and validate the DEM model. We discuss the coupling of the GRAnular Physics Engine (GRAPE) and the data-based models with different models of an excavator. Finally, we deploy the two realtime models from the previous Chapter 4 at the driving simulator RODOS, present several numerical studies, and discuss practical choices within the aforementioned models.

2 Mathematical Foundation

In this Chapter, we will introduce the most important mathematical concepts, which are needed throughout this work. To describe the trajectory of many particle systems in Chapter 3, we need certain concepts like basic Ordinary Differential Equation (ODE) theory, function spaces and multibody dynamics. For data-based models, developed in Chapter 4, we require concepts from the field of Supervised Learning, more specifically on Lookup Tables and Neural Networks.

First, we will introduce vector and relevant function spaces in Section 2.1 in order to fix the notation and for the description of particle systems. Then, we take a brief glance at ODEs and common numerical schemes in Section 2.2. Multibody Systems are covered in Section 2.5 as they play an important role, not only for the modeling of earthmoving machinery, but also as a concept to describe particle systems. Thereafter, we will make a short digression into stochastic calculus in Section 2.3, as some effort went into describing time series in the context of Lookup tables as a stochastic process in Chapter 4. In the succeeding Section 2.4, we cover the main aspects on Lookup Tables and Feedforward and Recurrent Neural Networks, both needed in Chapter 4. Finally, we present some error functions in Section 2.6, which will be needed in Chapter 5 to measure the prediction quality and accuracy of our models.

2.1 Function Spaces

Hilbert and Banach Spaces We recall that a *vector space* $\mathcal{V}$ over a field K, is a space with a mapping

$$+ : \mathcal{V} \times \mathcal{V} \longrightarrow \mathcal{V}$$

and a compatible mapping

$$\cdot : K \times \mathcal{V} \longrightarrow \mathcal{V}.$$

The K-vector space $\mathcal{V}$ forms an Abelian group with respect to $+$, the *vector addition*. The second mapping $\cdot$ is called *scalar multiplication*.

Definition 2.1.1. A norm is a mapping $\|\cdot\| : \mathcal{V} \longrightarrow K$, with the following properties.

- Non-negativity, i.e. for every $\boldsymbol{x} \in \mathcal{V} : \|x\| \geq 0$.
- Definiteness, i.e. $\|\boldsymbol{x}\| \Leftrightarrow \boldsymbol{x} = 0$.
- Compatibility with scalar multiplication, i.e. for $\boldsymbol{x} \in \mathcal{V}, \alpha \in$K: $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|$.
- Triangle inequality, i.e. for $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{V}$ it holds $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$.

A Banach space $(\mathcal{V}, \|\cdot\|_{\mathcal{V}})$ is a complete vector space $\mathcal{V}$ with a norm $\|\cdot\|_{\mathcal{V}}$. A Hilbert space $(\mathcal{V}, \langle\cdot,\cdot\rangle_{\mathcal{V}})$ is a complete vector space $\mathcal{V}$ with an inner product $\langle\cdot,\cdot\rangle_{\mathcal{V}}$.

The inner product naturally induces a norm $\|\cdot\|_{\mathcal{V}} = \sqrt{\langle\cdot,\cdot\rangle_{\mathcal{V}}}$. Thus every Hilbert space is a Banach space with respect to the induced norm.

The most commonly used example of a vector space is the $\mathbb{R}$-vector space $\mathbb{R}^n$ with the Euclidean norm $\|\cdot\|_2$. If it is clear from context, we usually omit the subscript $_2$ for better readability.

The following repetition of fundamental functional analytical results is based upon [Alt16, Chapter 3, pp. 37]. Let Y be a Banach space, and $\Omega \subset \mathbb{R}^n$.

Definition 2.1.2. The K-vector space of continuous functions is defined as the set

$$C^0(\Omega; Y) := \{f : \Omega \longrightarrow Y, f \text{ is continuous on } \Omega\}.$$

The following definition is more general and contains the previous one for $m = 0$.

Definition 2.1.3. For any integer $m \geq 1$, we denote the space of m-times differentiable functions as the set

$$C^m(\Omega; Y) := \{f : \Omega \longrightarrow Y \mid f \text{ is } m\text{-times coninuously differentiable in } \Omega\}.$$

Definition 2.1.4. A vector space $\mathcal{V}$ endowed with a K-bilinear mapping

$$\begin{aligned} \times_{\mathcal{V}} : \mathcal{V} \times \mathcal{V} &\longrightarrow \mathcal{V} \\ (\boldsymbol{a}, \boldsymbol{b}) &\longmapsto \boldsymbol{a} \times_{\mathcal{V}} \boldsymbol{b} \end{aligned}$$

which we call vector product. We call $\mathcal{V}$ an algebra.

Example 2.1.5. The vector space $\mathbb{R}^3$ with the cross product $\times_{\mathbb{R}^3}$ is an algebra.

Example 2.1.6. Every function space $\mathcal{F} \subset C^0(\mathbb{R}^m, \mathbb{R}^n)$ endowed with componentwise pointwise multiplication (also called *Hadamard product*)

$$\begin{aligned} \times_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} &\longrightarrow \mathcal{F} \\ (f,g)(\boldsymbol{x}) &\longmapsto \begin{pmatrix} f_1(\boldsymbol{x})g_1(\boldsymbol{x}) \\ \dots \\ f_n(\boldsymbol{x})g_n(\boldsymbol{x}) \end{pmatrix} \end{aligned}$$

is an algebra.

Let us first fix some notation in order to lose abstraction and be more concrete. We denote a column vector by $\boldsymbol{x}$ lying in the vector space $\mathbb{R}^n$ for some n in the natural numbers. More specifically, we consider $\boldsymbol{x}$ to be a matrix with only one column, i.e. $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$. The transpose of $\boldsymbol{x}$ is denoted by $\boldsymbol{x}^{\mathrm{T}}$ lying in the dual space $\mathbb{R}^{1\times n}$ of $\mathbb{R}^{n\times 1}$. As noted previously, the scalar product induces a norm $\langle \boldsymbol{x}, \boldsymbol{x}\rangle = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} \in \mathbb{R}$. We will often consider vector valued functions $\boldsymbol{x}(t) \in \mathcal{F}(\mathbb{R}_{\geq 0}, \mathbb{R}^n)$, depending on time t, which can be interpreted as curves in $\mathbb{R}^n$. Here, $\mathbb{R}_{\geq 0}$ describes the non-negative real numbers, i.e. the left-bounded interval $[0, \infty)$, and $\mathcal{F}$ some function space, e.g. $\mathcal{F} \subset C^m$. If it is clear from context, we may omit the parameter t, that is, we write $\boldsymbol{x}$ instead of $\boldsymbol{x}(t)$. For a numerical discretization of $\boldsymbol{x}(t)$ at time steps $t_0, \dots, t_i, \dots, t_f$, we will write $\boldsymbol{x}_i := \boldsymbol{x}(t_i)$. If we consider the norm of a vector $\|\boldsymbol{x}(t)\|_2$, the total time derivative is given by

$$\frac{d\|\boldsymbol{x}(t)\|_2}{\mathrm{d}t} = \frac{\langle \boldsymbol{x}(t), \dot{\boldsymbol{x}}(t)\rangle}{\|\boldsymbol{x}(t)\|_2}.$$

We will need this term in order to formulate a damping term in the Discrete Element Model in Chapter 3. Keeping these introductory notions in mind, we may define the following function spaces. Let X, Y be Banach spaces and $\Omega \subset X$ an open set.

Definition 2.1.7. A function $f : \Omega \to Y$ is called *Hölder-continuous* if for some $\alpha \in (0, 1]$ and for every $\boldsymbol{x}, \boldsymbol{y} \in \Omega$ it holds

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq C\|\boldsymbol{x} - \boldsymbol{y}\|^{\alpha}. \tag{2.1}$$

Let us recall the definition of Lipschitz continuity, we also refer to [Soh03, Chapter 4, p. 140].

Definition 2.1.8. A function $f : \Omega \to Y$ is called *globally Lipschitz-continuous* if it is Hölder continuous and $\alpha = 1$. That is

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq C\|\boldsymbol{x} - \boldsymbol{y}\| \quad \text{for all } \boldsymbol{x}, \boldsymbol{y} \in \Omega. \tag{2.2}$$

We say that f is *locally Lipschitz-continuous* if for $\boldsymbol{x} \in \Omega$, there exists an $\varepsilon > 0$ such that for all $\boldsymbol{y}$ in the ball $B_\varepsilon(\boldsymbol{x}) \cap \Omega$ it holds the Lipschitz condition. That is

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq C\|\boldsymbol{x} - \boldsymbol{y}\| \quad \text{for all } \boldsymbol{y} \in B_\varepsilon(\boldsymbol{x}) \cap \Omega. \tag{2.3}$$

The function space of all Lipschitz-continuous functions will be denoted by

$$Lip(\Omega; Y) := \{f : \Omega \longrightarrow Y \mid f \text{ is Lipschitz-continuous on } \Omega\}.$$

For the following definition, we refer to [Nat64, p. 243].

Definition 2.1.9. A function $f : [a, b] \to \mathbb{R}$ is said to be *absolutely continuous*, if for every $\varepsilon > 0$ there exists a $\delta > 0$, such that for a non-overlapping set of intervals $[t_i, t_{i+1}]$, with $\sum |t_{i+1} - t_i| \leq \delta$, we have $\sum |f(t_{i+1}) - f(t_i)| \leq \varepsilon$. The function space of absolutely continuous functions will be denoted by

$$AC([a, b]; \mathbb{R}) := \{f : [a, b] \longrightarrow \mathbb{R} \mid f \text{ is absolutely continuous on } [a, b]\}.$$

Definition 2.1.10. A function $f : [a, b] \to \mathbb{R}$ and any discretization $P = (x_k)_{k=0}^n$ with $a = x_0 < \cdots < x_n = b$ of the interval $[a, b]$, we set

$$V(f, P) := \sum_{j=1}^{n} |f(x_j) - f(x_{j-1})|.$$

Then we call

$$V_a^b(f) =: \sup\{V(f, P) : P \text{ is a partition of } [a, b]\} \tag{2.4}$$

the *total variation* of f. The function f is of *bounded variation*, if $V_a^b(f)$ is finite. The function space of functions with bounded variation will be denoted by

$$BV([a, b]; \mathbb{R}) := \{f : [a, b] \longrightarrow \mathbb{R} \mid f \text{ has bounded variation on } [a, b]\}.$$

Theorem 2.1.11. Let $f : \mathbb{R} \to \mathbb{R}$ be Lipschitz. Then it is also absolutely continuous.

Proof. Let $\varepsilon > 0$, and set $\delta = \varepsilon/C$. Then for any intervals $[x_i, y_i]$ with $\sum_i |x_i - y_i| \leq \delta$, we have that $\sum_i |f(x_i) - f(y_i)| \leq C \sum_i |x_i - y_i| < \delta$. □

2.2 Ordinary Differential Equations

Let us recall the notation of an ODE , which generally can be written implicitly as

$$F(t, \boldsymbol{y}, \dot{\boldsymbol{y}}) = 0,$$

where $\boldsymbol{y} : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}^n$ is the unknown. It becomes an Initial Value Problem (IVP) by adding the term

$$\boldsymbol{y}(t_0) = \boldsymbol{y}_0.$$

Locally, by the implicit function theorem we may write

$$\begin{aligned} \dot{\boldsymbol{y}} &= f(t, \boldsymbol{y}), \\ \boldsymbol{y}(t_0) &= \boldsymbol{y}_0, \end{aligned} \tag{2.5}$$

which corresponds to the explicit formulation.

Theorem 2.2.1 (Picard-Lindelöf)**.** Let $E \subset [t_0, T] \times \mathbb{R}^n$ and $f : E \to \mathbb{R}^n, (t, \boldsymbol{y}) \mapsto f(t, \boldsymbol{y})$ be continuous and locally Lipschitz continuous in the second variable $\boldsymbol{y}$, see also Definition 2.1.8. Then for every $(t^\star, c) \in E$ there exists an $\varepsilon \geq 0$ such that the Initial Value Problem (2.5) has a unique solution.

Proof. The proof can be found in [Arn73, Section 3.1, p. 221] or [For17, Section 12, p. 168]. □

Among all the numerical schemes designed to solve an ODE, the simplest one is the explicit or forward Euler method. Here we choose a not necessarily equidistant time grid

$$t_0, t_1 = t_0 + h_0, \ldots, t_N = T,$$

to discretize the interval $[t_0, T]$. Then, we start approximating $\boldsymbol{y}$ by $\tilde{\boldsymbol{y}}$ using

$$\tilde{\boldsymbol{y}}(t_1) = \boldsymbol{y}(t_0) + f(t_0, \boldsymbol{y}(t_0))h_0. \tag{2.6}$$

Or generally given the approximated solution $\tilde{\boldsymbol{y}}(t_i)$ of $\boldsymbol{y}(t_i)$ at time t_i, we compute

$$\tilde{\boldsymbol{y}}(t_{i+1}) = \tilde{\boldsymbol{y}}(t_i) + f(t_i, \tilde{\boldsymbol{y}}(t_i))h_i. \tag{2.7}$$

Usually, we consider an equidistant discretization with $h = h_i$ for all $i = 1, \ldots, N$. It is well-known, that the forward Euler is a first order method, which converges for a variety of problems given that the time step is small enough. Especially

stiff problems are often better treated using implicit methods. The implicit Euler requires a slight change in Equation (2.7), namely

$$\boldsymbol{y}(t_{i+1}) \approx \boldsymbol{y}(t_i) + f(t_{i+1}, \boldsymbol{y}(t_{i+1}))h_i. \tag{2.8}$$

The equation is only defined implicitly, because the unknown $\boldsymbol{y}(t_{i+1})$ also appears in f. This is more difficult to solve for one time step, but often worthwhile, as shown in [HW91, p. 2ff] and also in their great educational paper on numerical methods for ODEs [HL15]. There is a wide range of theory on ODEs and their numerical solution using higher order methods, which we will not cover here, because we will not address any of it in the upcoming chapters.

2.3 Stochastic Calculus

The following section is based upon [Eva12; KP99]. We aim to introduce the basic concepts regarding stochastic integrals to be able to formulate stochastic processes for a Stochastic Differential Equation (SDE). We need this for an improved version of the DEM Lookup approach, presented in Section 2.4.1.

Preliminaries Let $(\Omega, \mathcal{A}, P)$ be a probability space. Here, Ω corresponds to a *sample space* and $\mathcal{A}$ is a *σ-algebra*, that is:

- $\mathcal{A}$ consists of subsets of Ω.
- $\mathcal{A}$ contains the sample space Ω.
- If $A \in \mathcal{A}$ then $\mathcal{A}$ also contains its complement A^c.
- For $A_1, \cdots, A_n, \cdots \in \mathcal{A}$, the union $\cup_{j=1}^{\infty} A_j$ is contained in $\mathcal{A}$.

Furthermore, $P : \Omega \to \mathbb{R}$ describes a *probability measure* on Ω, namely

- $P(\Omega) = 1$ and $P(\emptyset) = 0$.
- If $A = \cup_{j=1}^{\infty} A_j$ for disjoint A_j, it holds that $P(A) = \sum_{j=1}^{\infty} P(A_j)$.

Definition 2.3.1. Let $\mathcal{A} = \mathcal{B}(\mathbb{R})$ denote the Borel σ-algebra. A function $X : \Omega \to \mathbb{R}$ is called *$\mathcal{A}$-measurable* if for every Borel-set $A \in \mathcal{B}(\mathbb{R})$, it hold that $X^{-1}(A)$ is contained in $\mathcal{A}$.

Definition 2.3.2. An $\mathcal{A}$-measurable function $X : \Omega \to \mathbb{R}$ is called a *random variable*.

Definition 2.3.3. If X is integrable, the expected value $\mathbb{E}[X]$ is defined as the Lebesgue integral over Ω, i.e.

$$\mathbb{E}[X] = \int_{\Omega} X(\omega) \mathrm{d}P(\omega).$$

Definition 2.3.4. A family of random variables $X = \{X_t, t \in [0,T]\}$, where the index set $[0,T]$ denotes an interval in $\mathbb{R}$ and can be considered as a time, is called a *stochastic process*.

Notation 2.3.5. A stochastic process X can be considered as a mapping

$$X : [0,T] \times \Omega \to \mathbb{R}, (t,\omega) \longrightarrow X(t,\omega) := X_t(\omega).$$

For $\omega \in \Omega$ we denote by

$$X^{\omega} : [0,T] \to \mathbb{R}, (t) \longrightarrow X^{\omega}(t,\omega) := X_t(\omega)$$

the sample path of $\omega \in \Omega$.

Construction of Brownian Motion

Definition 2.3.6. Let $\{W_t, t \in [0,T]\} = W$ be a stochastic process with

- $W_0 = 0$ almost surely.
- $W_t - W_s$ is normally distributed, i.e $W_t - W_s \sim \mathcal{N}(0, t-s)$,
- For $0 = t_1 < t_2 < \cdots < t_{n+1} = T$ the increments $W_{t_2} - W_{t_1}, \ldots, W_{t_{n+1}} - W_{t_n}$ are independent,
- W^{ω} is continuous for almost every $\omega \in \Omega$.

Then W is a Brownian motion, see Figure 2.1 for different sample paths of a Brownian motion.

For the simulation of a Brownian motion using the Euler-Maruyama scheme, it suffices to consider the following:

- $W_t - W_s \sim \mathcal{N}(0, t-s)$,
- $X_h \sim \mathcal{N}(0,1) \Rightarrow \sqrt{h} X_h \sim \mathcal{N}(0,h)$,
- $W_{t+h} \approx W_t + \sqrt{h} X_h$.

Definition 2.3.7. A familiy of sub-σ-algebras $\mathcal{A}_{t_0}, \ldots, \mathcal{A}_{t_n}, \cdots \subset \mathcal{A}$ with that property that for $0 \leq s \leq t$ $\mathcal{A}_s$ is contained in $\mathcal{A}_t$ is called a filtration of $\mathcal{A}$.

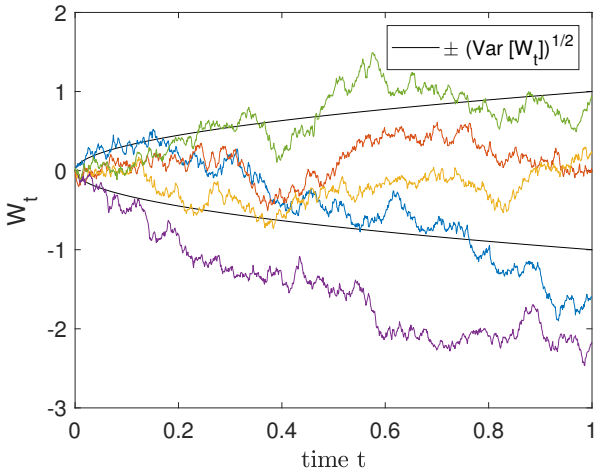


Figure 2.1: Different sample paths of a Brownian motion W. The square-root of the variance is displayed in black.

Definition 2.3.8. Let $\mathcal{A}_t \subset \mathcal{A}$ be a sub-σ-algebra of $\mathcal{A}$ and $X : \Omega \longrightarrow \mathbb{R}$ a random variable. $\mathbb{E}[X|\mathcal{A}_t]$ is the almost surely unique function from Ω to $\mathbb{R}$ satisfying

a) $\mathbb{E}[X|\mathcal{A}_t]$ is $\mathcal{A}_t$-measurable,

b) $\int_A X \mathrm{d}P = \int_A \mathbb{E}[X|\mathcal{A}_t]\mathrm{d}P$ for all sets $A \in \mathcal{A}_t$.

The random variable $\mathbb{E}[X|\mathcal{A}_t]$ is called the conditional expectation of X with respect to $\mathcal{A}_t$.

Remark 2.3.9. The conditional expectation has the following properties:

a) $\mathbb{E}[\mathbb{E}[X|\mathcal{A}_t]] = \mathbb{E}[X]$,

b) $\mathbb{E}[X|\mathcal{A}_t] = X$ if X is $\mathcal{A}_t$-measurable, i.e. for all $A \in \mathcal{B}(\mathbb{R}) : X^{-1}(A) \in \mathcal{A}_t$.

2.4 Supervised Learning

Supervised Learning describes the general procedure of learning a function mapping input data to output data. Training data consists of a relation of some input $\mathcal{I} \subset \mathbb{R}^m$ and some output $\mathcal{O} \subset \mathbb{R}^n$. Here, we assume N tupels $(\boldsymbol{u}_k, \boldsymbol{y}_k) \in \mathcal{I} \times \mathcal{O}$ for $k = 1, \ldots, N$. A supervised learning algorithm uses the training data to generate

an inferred function. This function may be written as

$$\begin{aligned} f : \mathbb{R}^m &\longrightarrow \mathbb{R}^n, \\ \boldsymbol{u} &\longmapsto \boldsymbol{y} := f(\boldsymbol{u}). \end{aligned} \tag{2.9}$$

The general procedure in supervised learning goes as follows. First, we analyze the relevant data and gather a training set by measurement or simulation. Second, we choose suitable input feature data, which characterizes the input data. This should be as small as possible but large enough to capture all features. Then a suitable learning algorithm can be selected. In this work, we focused on LUTs and on RNNs. Thereafter, we perform the model generation by applying the learning algorithm. Finally, we test the model with data, ideally not included in the training data set.

2.4.1 Lookup Tables

The general idea of Lookup Tables is old and has been used for several hundred years for example in the manual computation of logarithms or sines, [LW92; Bür20]. If we want to evaluate a possibly nonlinear function in an efficient way, we precompute data at specific points in a previous offline phase. These function values are stored in a data structure - in previous centuries a table - and can be accessed efficiently while for points in between some interpolation routine can be applied. Logarithm tables have been in use in school education in Europe until the introduction of the first electronic calculators in the early 70s.

Therefore, let us consider Equation (2.9), where f is a possibly nonlinear function. For $\boldsymbol{u} \in \mathbb{R}^m$, we perform a Taylor approximation at $\boldsymbol{u} = \boldsymbol{u}_i + \Delta\boldsymbol{u}$:

$$f(\boldsymbol{u}) = f(\boldsymbol{u}_i) + \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{u}}|_{\boldsymbol{u}_i}(\boldsymbol{u} - \boldsymbol{u}_i) + \mathcal{O}((\boldsymbol{u} - \boldsymbol{u}_i)^2). \tag{2.10}$$

Several approximation routines can then be stated, e.g. the definitions in [Her08, Chapter 4, p. 61] and [Kre14, Chapter 1.3.3, p. 11f]. Let us therefore prescribe an equidistant discretization of a hypercube $\mathcal{P} \subset \mathbb{R}^m$, given by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$.

Definition 2.4.1 (Lookup 0)**.** The *Lookup 0* approximation of the function f at point $\boldsymbol{u}$ is given at the closest point $\tilde{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_N} \|\boldsymbol{u} - \boldsymbol{u}_i\|_2$. Then the approximation yields

$$f(\boldsymbol{u}) \approx f(\tilde{\boldsymbol{u}}).$$

Definition 2.4.2 (Lookup 1)**.** The *Lookup 1* approximation of the function f at point $\boldsymbol{u}$ is given at the closest point $\tilde{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_N} \|\boldsymbol{u} - \boldsymbol{u}_i\|_2$. Then the approximation yields

$$f(\boldsymbol{u}) \approx f(\tilde{\boldsymbol{u}}) + \frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{u}}|_{\tilde{\boldsymbol{u}}}(\boldsymbol{u} - \tilde{\boldsymbol{u}}).$$

Definition 2.4.3 (TPWL)**.** The Trajectory Piecewise Linear (TPWL) approximation of a function f at a point $\boldsymbol{u}$ is defined by the k closest points $\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_k$. These can be found by successively computing the closest point of the set $\mathcal{P} \setminus \{\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_\ell\}$ for $\ell = 1, \ldots, k$. Then we obtain the sequence of closest points $\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_k$ with

$$\|\boldsymbol{u} - \tilde{\boldsymbol{u}}_1\|_2 \leq \|\boldsymbol{u} - \tilde{\boldsymbol{u}}_2\|_2 \leq \ldots \|\boldsymbol{u} - \tilde{\boldsymbol{u}}_k\|_2.$$

Then the TPWL approximation is given by

$$f(\boldsymbol{u}) \approx \sum_{i=1}^{k} \omega_i f(\tilde{\boldsymbol{u}}_i),$$

for suitable unit weights, that is $\sum_{i=1}^{k} \omega_i = 1$.

The right choice of the weights is discussed in the following. In the original work of Rewieński [Rew03], they propose to use exponentially decaying weights based on the Euclidean Distance

$$d_\ell = \|\boldsymbol{u} - \tilde{\boldsymbol{u}}_\ell\| \text{ for } \ell = 1, \ldots, k.$$

With the above construction, $d_1 = \min_{1,\ldots,k} d_\ell$ is the minimum distance of the nearest neighbor. We set the weights $\tilde{\omega}_\ell = \exp(-\beta d_\ell / d_1)$ and scale them to unit length by

$$\omega_\ell = \frac{\tilde{\omega}_\ell}{\sum_{i=1}^{k} \tilde{\omega}_i}.$$

The artificial parameter β is set to the value 25 in [Rew03, Chapter 3, p. 41]. If it is reduced, the closer points gain more influence and vice versa.

We consider a different approach for the weights. For all points $\boldsymbol{u}_i \in \mathcal{P} \subset \mathbb{R}^m$, we assume a normal Gaussian distribution and compute the standard deviation $\sigma_{\mathcal{P}}$. The standardized Euclidean distance is defined by

$$d_\ell = \frac{\|\boldsymbol{u} - \boldsymbol{u}_\ell\|}{\sigma_{\mathcal{P}}} \text{ for } \ell = 1, \ldots, k.$$

We compute the weights by normalizing the standardized Euclidean distance

$$\omega_\ell = \frac{d_\ell}{\sum_{i=1}^{k} d_i}.$$

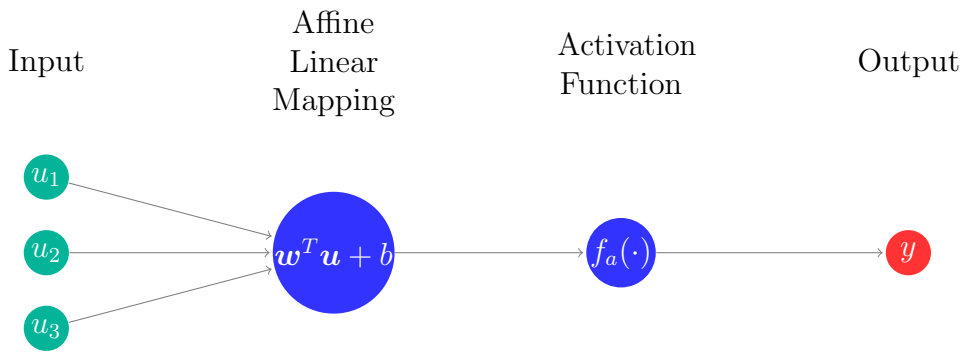


Figure 2.2: Visualization of a single neuron of an artificial Neural Network.

The Lookup 0 approach is a simple form of interpolation of a multi-dimensional function. Lookup 1 incorporates the derivative at point $\tilde{\boldsymbol{u}}$. With a sufficiently smooth function f, we expect convergence in the order $\mathcal{O}(h)$.

The TPWL approach incorporates more neighboring points, but is also a pure interpolation routine. Following the idea of Taylor expansion, by including higher derivatives, we increase the order of convergence, given that the approximated function is sufficiently smooth.

2.4.2 Feedforward Neural Networks

Feedforward Neural Networks (FNNs) consist of at least one input layer and an output layer, several hidden layers may lie in between. Each layer consists of nodes, also called neurons. A neuron consists of a weight vector $\boldsymbol{w}$ and a bias $\boldsymbol{b}$, and an activation function $f_a(\cdot)$, see Figure 2.2. Several neurons form a layer in the Neural Network. We may collect the weights to form a weight matrix W. Furthermore, each layer comprises a typically nonlinear activation function f_a. The input $\boldsymbol{u}$ is then processed to compute the output $\boldsymbol{y} = N(\boldsymbol{u})$, where N is the nonlinear function defined by the neural network. The training is typically based on back propagation, adjusting the weights and biases in the nodes successively, starting at the output layer [RHW86; LeC+12]. The trained network is called a FNN, if the nodes do not form cycles. More specifically, the information passes only in one direction, without influence of previous time steps. FNNs can approximate any continuous function. We introduce the main ingredients to formulate Theorem 2.4.11 of Universal Approximation for FNNs. The proof is by Kurt Hornik [HSW89] and relies upon the Stone-Weierstrass theorem. The following definition is by [SZ06].

Definition 2.4.4. A real-valued function $f_\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ is called a sigmoid, if f_σ is monotonically increasing and if it is bounded. That is for $a < b$, it holds $f_\sigma(a) < f_\sigma(b)$,

$$f_\sigma(a) \in [\alpha, \beta], \quad \lim_{a \to -\infty} f_\sigma(a) = \alpha \text{ and } \lim_{b \to \infty} f_\sigma(b) = \beta.$$

Usually, we set $\alpha = 0$ or $\alpha = -1$ and $\beta = 1$.

Let us recall the Definition 2.1.4 of an algebra from Chapter 2. We consider subsets $\mathcal{A} \subset \mathcal{C}^I$ which are closed with respect to addition and scalar multiplication.

Definition 2.4.5. An Algebra $\mathcal{A}$ *vanishes on no point of a compact set* $K \subset \mathbb{R}^I$ if for every $\boldsymbol{x} \in K$, there exists an $f \in \mathcal{A}$ such that $f(\boldsymbol{x}) \neq 0$.

Definition 2.4.6. An Algebra $\mathcal{A}$ *separates points on a compact set* $K \subset \mathbb{R}^I$ if for pairwise different $\boldsymbol{x}, \boldsymbol{y} \in K, \boldsymbol{x} \neq \boldsymbol{y}$ there exists an $f \in \mathcal{A}$ such that $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$.

Now, we are able to formulate the main ingredient in the proof of Theorem 2.4.11.

Theorem 2.4.7 (Stone-Weierstrass)**.** Let $\mathcal{A}$ be an algebra of real continuous functions on a compact set K. If $\mathcal{A}$ separates points on K and if $\mathcal{A}$ vanishes at no point on K, then $\mathcal{A}$ lies ρ_K-dense in the space of real continuous functions.

Proof. The proof can be found in [Soh03, Chapter 9.4, p. 383]. □

Definition 2.4.8. The class of all Lebesgue measurable functions from $\mathbb{R}^I$ to $\mathbb{R}^n$ is denoted by $\mathcal{M}^{I,n} = \mathcal{M}(\mathbb{R}^I; \mathbb{R}^n)$.

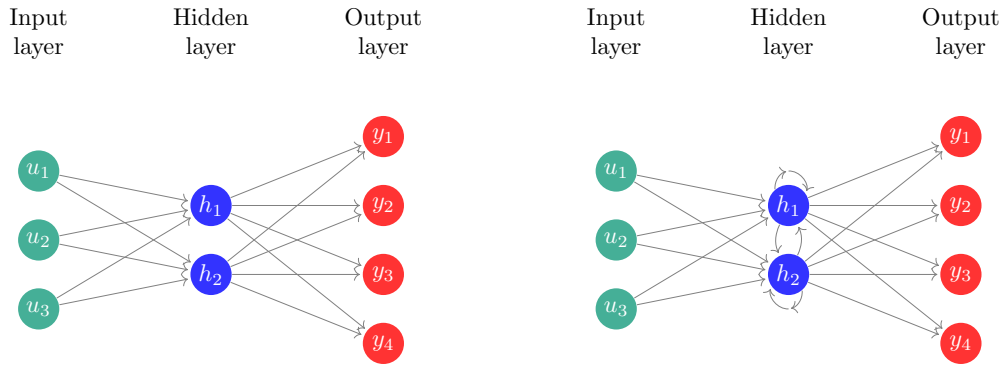
Definition 2.4.9. Let A^I with $I \in \mathbb{N}$ be the set of all affine mappings

$$A : \mathbb{R}^I \longrightarrow \mathbb{R}, \boldsymbol{x} \longmapsto A(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle - \psi,$$

where $\boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^I$ and $\psi \in \mathbb{R}$.

Definition 2.4.10. For a Borel-measurable function $f = (f_1, \ldots, f_n)^T : \mathbb{R} \longrightarrow \mathbb{R}^n \in \mathcal{M}^{1,n}$, we set $\Sigma^{I,n}(f) = \left\{ NN : \mathbb{R}^I \longrightarrow \mathbb{R}^n \mid NN(\boldsymbol{x}) = \sum_{j=1}^{J} V_j f(A_j(\boldsymbol{x})) \right\}$. Here, $\boldsymbol{x} \in \mathbb{R}^I$ and $V_j \in \mathbb{R}^{1 \times n}$, where the product between V_j and f is the Hadamard product, i.e. we multiply both column vectors componentwise. The affine mapping A_j belongs to A^I and $J \in \mathbb{N}$.

The function NN corresponds to a Neural Network with one hidden layer. We can now state the Theorem on Universal Approximation.



(a) Feedforward Neural Network

(b) Recurrent Neural Network

Figure 2.3: Visualization of a FNN and a RNN

Theorem 2.4.11 (Universal Approximation of FNN)**.** Let f be a non-constant continuous function $f : \mathbb{R} \mapsto \mathbb{R}^n$. Then $\Sigma^{I,n}(f)$ is uniformly dense on compacta in $C(\mathbb{R}^I; \mathbb{R}^n)$.

Proof. See Appendix A.2. □

As a consequence, we may state that every continuous function $g : \mathbb{R}^I \to \mathbb{R}^n$ can be approximated to arbitrary accuracy by a function in $\Sigma^{I,n}(f)$.

2.4.3 Recurrent Neural Networks

A RNN contains cycles between the hidden nodes. That is, the output of a hidden node of a previous time step serves as input of the hidden nodes of the next time step, see Figure 2.4. The idea for this illustration comes from [GBC16, Figure 10.3, p. 373]. This time-dependent behavior is suitable to mimic a discrete dynamical system. Let us recall that a discrete dynamical system with input $\boldsymbol{u}$ and output $\boldsymbol{y}$ may be written as

$$\begin{aligned} \boldsymbol{s}_{t+1} &= f(\boldsymbol{s}_t, \boldsymbol{u}_t), \\ \boldsymbol{y}_t &= h(\boldsymbol{s}_t). \end{aligned} \tag{2.11}$$

The first part of Equation (2.11) is called the state transition. The second part refers to the output equation [SZ06]. It can be shown that every dynamical system can be approximated by a Recurrent Neural Network up to arbitrary accuracy. It is

possible to prove that for every $\varepsilon > 0$ and for any given input $\boldsymbol{u}$ and any given output $\boldsymbol{y}$, there exists a Recurrent Neural Network NN, such that $\|NN(u) - \boldsymbol{y}\| < \varepsilon$. This property is called universal approximation. Note that we did not further specify the net architecture or size. The proof is based upon Schäfer, see [SZ06], which builds upon Theorem 2.4.11 by Hornik, see [HSW89]. We recapitulate their proof in the appendix.

Definition 2.4.12. For any Borel-measurable function $f(\cdot) : \mathbb{R}^J \mapsto \mathbb{R}^J$, we define the class $RNN(f)$ by functions of the form

$$\begin{aligned} \boldsymbol{s}_{t+1} &= f(A\boldsymbol{s}_t + B\boldsymbol{u}_t - \psi), \\ \boldsymbol{y}_t &= C\boldsymbol{s}_t. \end{aligned} \tag{2.12}$$

Theorem 2.4.13 (Universal Approximation of RNN)**.** Let us consider a discrete dynamical system

$$\begin{aligned} \boldsymbol{s}_{t+1} &= g(\boldsymbol{s}_t, \boldsymbol{u}_t), \\ \boldsymbol{y}_t &= h(\boldsymbol{s}_t), \end{aligned} \tag{2.13}$$

where $g : \mathbb{R}^J \times \mathbb{R}^J \mapsto \mathbb{R}^J$ is Borel-measurable and $h : \mathbb{R}^J \mapsto \mathbb{R}^n$ is continuous. Then there exists an element of $RNN(f)$ which approximates the dynamical system given by Equation (2.13).

Proof. See Appendix A.2 or [SZ06]. □

The universality property is a key ingredient for the theoretical ability of FNNs. Given enough hidden nodes, it suffices to train a network with one hidden layer and approximate any continuous function with arbitrary accuracy. A further powerful property, which we just want to mention here, is that Recurrent Neural Networks are - in a certain sense - Turing complete [SS92].

Scaling of Input and Output Data From a practical point of view, it is absolutely advisable to scale the training data. In the following, we want to share our experience and describe different aspects of scaling input and output data. First, we consider a relatively general scaling operation. Thereafter, we present two scaling variants depending on the statistical distribution of the data.

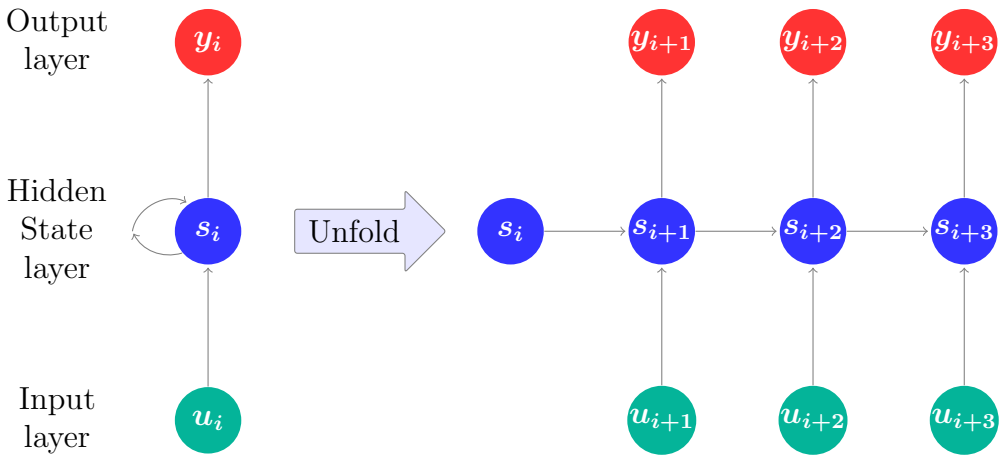


Figure 2.4: Visualization of the recurrent structure of an RNN. The current state serves as input for the hidden state of the coming time step.

Scaling Variant 0: Let us assume that we have collected sufficient training data $(\boldsymbol{u}_k, \boldsymbol{y}_k)$ for $k = 1, \ldots, N$. Here, the index k can be understood as a time discretization. Then we may compute for each input variable i for $i = 1, \ldots, n$ the maximum $M(i) = \max_{k=1,\ldots,N} \boldsymbol{u}_k(i)$ and the minimum $m(i) = \min_{k=1,\ldots,N} \boldsymbol{u}_k(i)$. Here, i denotes the index of the input dimension, and k the index for a time step. Then, the scaling function

$$\begin{aligned} S^i : [m(i), M(i)] &\longrightarrow [-1, 1], \\ \boldsymbol{u}(i) &\longmapsto \tilde{\boldsymbol{u}}(i) := -1 + \frac{\boldsymbol{u}(i) - m}{M - m}(1 - (-1)), \end{aligned}$$

maps all data to the interval $[-1, 1]$. The inverse scaling function is given by

$$\begin{aligned} \left(S^i\right)^{-1} : [-1, 1] &\longrightarrow [m(i), M(i)] \\ \tilde{\boldsymbol{u}}(i) &\longmapsto m + \frac{\tilde{\boldsymbol{u}}(i) + 1}{1 - (-1)}(M - m). \end{aligned}$$

This ensures that all input data lie in the hyper cube $[-1, 1]^n$. Analogously, we can proceed with the output data. Both transformations ensure that back propagation converges faster, but have the drawback of the requirement to compute the minima and maxima.

Scaling Variant 1: A different scaling mapping can be defined by assuming a normal distribution in the data. For each input column, we compute the expected

value over all training data by $\mathbb{E}[\boldsymbol{u}(i)] = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{u}_k(i)$ and the standard deviation $\sigma[\boldsymbol{u}(i)] = \frac{1}{N}\sum_{k=1}^{N}((\boldsymbol{u}_k(i) - \mathbb{E}[\boldsymbol{u}(i)])^2)$. Then rescaling is performed for all $k = 1, \ldots, N$ by

$$\tilde{\boldsymbol{u}}_k(i) := \frac{\boldsymbol{u}_k(i) - \mathbb{E}[\boldsymbol{u}(i)]}{\sigma[\boldsymbol{u}(i)]}.$$

The analogous scaling works also for the output data. Again, we need to store the expected value and the standard deviation in the input data for each $i = 1, \ldots, n$ and in the output data for each $i = 1, \ldots, m$.

Scaling Variant 2: Another option is less physical and ignores the fact, that the input columns may have different physical units. For each time step k, we compute the expectation over all inputs i, namely $\mathbb{E}[\boldsymbol{u}(k)] := \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{u}_k(i)$ and the standard deviation $\sigma[\boldsymbol{u}(k)] = \frac{1}{n}\sum_{i=1}^{n}((\boldsymbol{u}_k(i) - \mathbb{E}[\boldsymbol{u}(k)])^2)$. The input is then scaled by

$$\tilde{\boldsymbol{u}}_k(i) := \frac{\boldsymbol{u}_k(i) - \mathbb{E}[\boldsymbol{u}(k)]}{\sigma[\boldsymbol{u}(k)]}$$

in each time step $k = 1, \ldots, N$. The advantage of this scaling operation is that storing any processed input data is not necessary. A drawback is the loss of physical interpretability of the data. This approach is closer to a black box model, where we disregard a physical explanation of the data.

Training and Data Collection In the context of excavation, we record the motion of the bucket while digging at a driving simulator. That way, we can generate plausible input trajectories. The trajectories consist of position $\boldsymbol{x}$ and current orientation $\boldsymbol{p}$, i.e. in generalized coordinates $\boldsymbol{q} = (\boldsymbol{x}, \boldsymbol{p})$, as well as their derivatives $\tilde{\dot{\boldsymbol{q}}} = (\boldsymbol{v}, \boldsymbol{\omega})$. As for the DEM Lookup approach, see Section 4.3.1, we start with an offline simulation of the full DEM model. That maps the bucket trajectory to the force torque output, i.e. we obtain our data $(\boldsymbol{u}^T, \boldsymbol{y}^T)^T$. We divide the data into training and test data.

Overfitting A common problem when using Neural Networks is overfitting. During training, the weights and biases are updated in order to minimize the distance between the function f from Equation (2.9) and the approximated function N. But after sufficient iterations, this can lead to the behavior that the training data is approximated correctly, while the error on the test data increases. We can coin this in terms of the following definition

Definition 2.4.14. Let $(U, Y) \subset \mathcal{I} \times \mathcal{O}$ be training data and $(\tilde{U}, \tilde{Y})$ test data and ε. A neural net $N : \mathcal{I} \longrightarrow \mathcal{O}$ is overfitted to (U, Y) if there exists a neural net N_2 such that

$$\begin{aligned} \varepsilon(N(U), Y) &\leq \varepsilon(N_2(U), Y), \\ \text{but } \varepsilon(N(\tilde{U}), \tilde{Y}) &> \varepsilon(N_2(\tilde{U}), \tilde{Y}). \end{aligned}$$

Simulation in Realtime Let us state a formal definition of realtime capable simulation or an algorithm.

Definition 2.4.15. Consider a time series in the time interval $[t_0, T]$ and a simulation routine describing the time series. If the execution time τ_s of the simulation is strictly smaller then the length τ_p of the interval $[t_0, T]$ for all time series, the simulation routine is realtime capable. The realtime factor r_f is defined as

$$r_f = \frac{\tau_s}{\tau_p}.$$

2.5 Multibody Systems

The following is based upon [Woe16]. A Multibody System (MBS) consists of N_B flexible or rigid bodies, described by masses and moments of inertia. Here, we focus only on rigid multibody systems. Furthermore, the bodies are connected by N_J joints and N_F force elements. The total number of degrees of freedom is denoted by N_D. We distinguish between open multibody systems, which can be described as systems in chain or in tree structure and closed systems, see Figure 2.5. Closed systems lead to a system of equations in form of a Differential Algebraic Equation (DAE), while open systems can be described by an ODE. The motion of a multibody system can be constrained. Such a constraint

$$\boldsymbol{g} = 0 \tag{2.14}$$

is called *holonomic* if it depends on the generalized coordinates $\boldsymbol{q}$, i.e. $\boldsymbol{g}(\boldsymbol{q}(t)) = 0$. A constraint is *rheonomic* if it explicitly depends upon time t, i.e. $\boldsymbol{g}(\boldsymbol{q}(t), t) = 0$. Otherwise, if the constraint does not explicitly depend upon time t, it is called *scleronomic*. The equality in Equation (2.14) marks a two-sided or *bilateral* constraint. If we can replace the equality in Equation (2.14) by an equality, i.e. $\boldsymbol{g} \geq 0$, we have a one-sided or *unilateral* constraint.

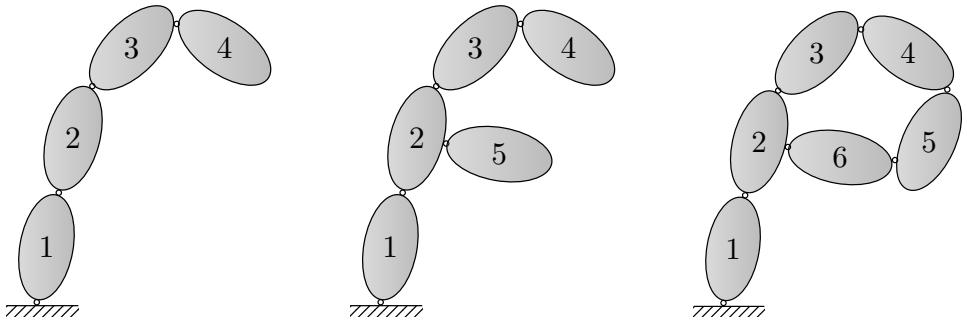


Figure 2.5: From left to right: sketch of open multibody systems in chain or tree structure or a closed multibody system

Rigid Bodies A rigid body may translate, corresponding to 3 degrees of freedom, and rotate, corresponding to another 3 degrees of freedom. It is further constrained by joints, which might reduce the total number of degrees of freedom.The bodies are assumed to be rigid. A rigid body is located in a global coordinate frame $\mathcal{I} = \{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}$. We may assign a body-fixed coordinate frame $\mathcal{K} = \left\{\boldsymbol{e}_x^K, \boldsymbol{e}_y^K, \boldsymbol{e}_z^K\right\}$ to each body. A coordinate frame consists of a rotation matrix $\mathcal{R}^{\mathcal{I}\mathcal{K}}$ and a point $p^{\mathcal{I}}$ with respect to the global coordinate frame. We usually set $p^{\mathcal{I}}$ in the center of gravity of a body.

System State Each rigid body is defined by its position $\boldsymbol{x} \in \mathbb{R}^3$. Furthermore, we need to account for the rotation of each body with respect to the global reference frame. This can be achieved by defining three angles $\boldsymbol{\theta} = (\alpha, \beta, \gamma)$, which may result in ambiguities as the rotation sequence with respect to the three principal axes has to be known. Another possibility is the use of rotation matrices, but here singularities may occur. Therefore, in all numerical experiments we will use unit quaternions $\boldsymbol{p} = (p_1, p_2, p_3, p_4)$, which can be understood as a 3-dimensional submanifold of $\mathbb{R}^4$. Unit quaternions and their properties are explained in more detail in the Appendix A.1. In this context, however, we will focus on angles $\boldsymbol{\theta}$, which eases the notation with respect to inertia tensors. To summarize, we present translational, rotational and general coordinates of kinematic units of interest in Table 2.1.

With a slight abuse of notation, we will not distinguish in this thesis between the $7N$ dimensional vector $\tilde{\boldsymbol{q}}$ and the $6N$ dimensional vector $\boldsymbol{q}$. The reason is, that we prefer to work with quaternions on a position level. On a velocity level, it is often more intuitive to work with angular velocities ω, resulting in the general velocities $\dot{\boldsymbol{q}}$. Also when computing the product between a mass matrix $M \in \mathbb{R}^{6\times 6}$

kinematic unit	translational	rotational	general	
dimension	3	3	6	7
position	$\boldsymbol{x}$	$\boldsymbol{\theta}$	$\boldsymbol{q} = (\boldsymbol{x}, \boldsymbol{\theta})$	$\tilde{\boldsymbol{q}} = (\tilde{\boldsymbol{x}}, \boldsymbol{p})$
velocity	$\boldsymbol{v}$	$\boldsymbol{\omega}$	$\dot{\boldsymbol{q}} = (\boldsymbol{v}, \boldsymbol{\omega})$	$\dot{\tilde{\boldsymbol{q}}} = (\boldsymbol{v}, \dot{\boldsymbol{p}})$
acceleration	$\boldsymbol{a}$	$\boldsymbol{\alpha}$	$\ddot{\boldsymbol{q}} = (\boldsymbol{a}, \boldsymbol{\alpha})$	$\ddot{\tilde{\boldsymbol{q}}} = (\boldsymbol{a}, \ddot{\boldsymbol{p}})$
jerk	$\boldsymbol{j}$	$\boldsymbol{\zeta}$	$\dddot{\boldsymbol{q}} = (\boldsymbol{j}, \boldsymbol{\zeta})$	$\dddot{\tilde{\boldsymbol{q}}} = (\boldsymbol{j}, \dddot{\boldsymbol{p}})$

Table 2.1: Kinematic notation and translational, rotational and general coordinates

and a generalized vector $\tilde{\boldsymbol{q}}$, it is more convenient to write $\boldsymbol{q}$. On the position level, we prefer to use quaternions, as they avoid any singularities and write $\tilde{\boldsymbol{q}}$.

Euler-Lagrange Formalism Let us consider a dynamical system with kinetic energy $\boldsymbol{T}$ and potential energy $\boldsymbol{V}$. The kinetic energy $\boldsymbol{T}$ depends upon the mass matrix $M(\boldsymbol{q})$ and the generalized velocities $\dot{\boldsymbol{q}}$, so we may write

$$\boldsymbol{T} = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}.$$

The external forces

$$\boldsymbol{f}_{ext} = -\frac{\partial \boldsymbol{V}}{\partial \boldsymbol{q}}. \tag{2.15}$$

depend upon the potential $\boldsymbol{V}$. Let us consider the gravitational force, given by $\boldsymbol{V} = Mg\boldsymbol{q}_z$, where $\boldsymbol{q}_z$ denotes the vector with entries in the third translational component

$$\boldsymbol{q}_z := (\boldsymbol{0}_2, q(3), \boldsymbol{0}_3, \ldots, \boldsymbol{0}_2, q(6(N-1)+3), \boldsymbol{0}_3).$$

Then we can formulate the Lagrange equation as

$$\mathcal{L} = \boldsymbol{T} - \boldsymbol{V}.$$

We obtain the equations of motion by referring to the Lagrange-formalism, namely considering

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = 0.$$

The first part yields

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} &= M(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \text{thus} \quad \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} &= \frac{dM(\boldsymbol{q})}{dt}\dot{\boldsymbol{q}} + M(\boldsymbol{q})\ddot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + M(\boldsymbol{q})\ddot{\boldsymbol{q}} \end{aligned}$$

Secondly, by definition we obtain

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \frac{\partial \boldsymbol{V}}{\partial \boldsymbol{q}},$$

leading to

$$\begin{aligned} \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} &= \dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + M(\boldsymbol{q})\ddot{\boldsymbol{q}} - \frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{V}}{\partial \boldsymbol{q}} \\ &= M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{V}}{\partial \boldsymbol{q}}. \end{aligned}$$

It can be shown, that the term $-\frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}$ corresponds to Coriolis and centrifugal forces. In the case of body coordinates of discs or spheres, the moment of inertia I_b is constant in time and the term vanishes. By introducing the total force

$$\boldsymbol{f} = -\frac{1}{2}\dot{\boldsymbol{q}}^T \frac{\partial M(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{f}_{ext},$$

we arrive at Newton's equations

$$M\ddot{\boldsymbol{q}} = \boldsymbol{f}.$$

Additional algebraic constraints

$$g(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = 0$$

lead to reaction forces, which we add in form of a term $G^T(\boldsymbol{q})\boldsymbol{\lambda}(t)$ on the right-hand side of the first equation, where $G = \frac{\partial g}{\partial \boldsymbol{q}}$. Summing up, we write

$$\begin{aligned} \boldsymbol{M}\ddot{\boldsymbol{q}} &= \boldsymbol{f} + G^T(\boldsymbol{q})\boldsymbol{\lambda}(t) \\ \boldsymbol{g}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) &= 0. \end{aligned}$$

Nonsmooth or penalty based Lagrange multipliers The Lagrange multiplier $\boldsymbol{\lambda}(t)$ may be computed as measures in order to enforce almost perfectly rigid collisions. On the other hand, in the penalty-based formulation, we take $\boldsymbol{\lambda}_{ij}(t) = k_n\delta_{ij} + d_n\dot{\delta}_{ij}$ where $\delta_{ij} = -g(\boldsymbol{q}_i, \boldsymbol{q}_j)$, see also [SRS00, p. 327].

Joints To connect two bodies i and j with a joint, we need to specify two additional points $P_i, \mathcal{K}_i^a$ and $P_j, \mathcal{K}_j^a$ on body i and j respectively. The rotation between $\mathcal{K}_i^a$ and $\mathcal{K}_i$ is constant, as both frames are on the respective rigid body.

A kinematic joint constrains two rigid bodies in a specific way. The number of constraints b and the number of Degrees of Freedom (DOF) f, equals 6, that is

$$f + b = 6.$$

A joint G_{ij} is defined implicitly by an equation of the form

$$g_{ij}(\boldsymbol{q}_i, \boldsymbol{q}_j, t) = 0.$$

Here the function

$$g_{ij} : \mathbb{R}^7 \times \mathbb{R}^7 \times \mathbb{R} \longrightarrow \mathbb{R}^{b_{ij}}$$

is assumed to be smooth, where b_{ij} describes the number of constraints.

2.6 Error Measures

Aiming at the correct prediction of time series, we need to define a suitable error measure. We want this to be independent of the trajectory of the tool, the time discretization and in addition to express meaningful information on the quality of approximation. Therefore, we define the following error functions. Let y be the reference solution and $\tilde{y}$ some approximation of y. For vector valued time series $\boldsymbol{y}$, we consider each component separately. We assume an equidistant time step $\Delta t = t_i - t_{i-1}$ for all $i = 2, \ldots, N$.

Definition 2.6.1. We define the following three error measures. The maximum error

$$\varepsilon_{\infty}^{a}(y, \tilde{y}) = \max_{t\in[t_0,t_N]} |y(t) - \tilde{y}(t)| \approx \max_{i=1,\ldots,N} |y(t_i) - \tilde{y}(t_i)|$$

describes the maximal distance of a time series from another. The $\mathcal{L}^2$ error

$$\varepsilon_{\mathcal{L}^2}^{a}(y, \tilde{y}) = \sqrt{\int_{t_0}^{t_N} |y(t) - \tilde{y}(t)|^2 \,\mathrm{d}t} \approx \sqrt{\sum_{i=1,\ldots,N} |y(t_i) - \tilde{y}(t_i)|^2 \,\Delta t}$$

is an error function, in an integral sense, where larger distances have a greater influence. The $\mathcal{L}^1$ error

$$\varepsilon^a_{\mathcal{L}^1}(y,\tilde{y}) = \int_{t_0}^{t_N} |y(t) - \tilde{y}(t)| \,\mathrm{d}t \approx \sum_{i=1,\ldots,N} |y(t_i) - \tilde{y}(t_i)| \,\Delta t$$

is another integral error function, which sums up all absolute errors linearly over time.

To compare different meneuvers and different time scales, a relative error is more suitable to measure the prediction quality of an approximation routine. The following definition, based on the aforementioned absolute error, serves the computation of relative errors with respect to the solution y.

Definition 2.6.2. The relative maximum error

$$\varepsilon^r_{\infty}(y,\tilde{y}) = \frac{\varepsilon^a_{\infty}(y,\tilde{y})}{\varepsilon^a_{\infty}(y,0)}$$

accounts to the maximum error with respect to the solution y. Similarly, the relative $\mathcal{L}^2$ error may be written as

$$\varepsilon^r_{\mathcal{L}^2}(y,\tilde{y}) = \frac{\varepsilon^a_{\mathcal{L}^2}(y,\tilde{y})}{\varepsilon^a_{\mathcal{L}^2}(y,0)}.$$

The relative $\mathcal{L}^1$ error reads

$$\varepsilon^r_{\mathcal{L}^1}(y,\tilde{y}) = \frac{\varepsilon^a_{\mathcal{L}^1}(y,\tilde{y})}{\varepsilon^a_{\mathcal{L}^1}(y,0)}.$$

3 Particle Models for Soil Simulation

In the previous chapter, we laid the theoretical and mathematical headstone for the remainder of this thesis. In this chapter, we want to focus on three different particle models and underline their benefits and weaknesses. We follow two goals in this chapter. First, we want to assess the possibility of constructing a real-time capable particle model suitable for force prediction. Second, we compare the different models regarding the generation of offline data, for a machine learning model. Here, we value accuracy over real-time capability, while still requiring an efficient simulation.

3.1 Introduction to Particle Methods

Soil simulation based on particles is a reliable proven physical modeling approach. The drawback is that for higher number of particles the computation time, even using CPU or GPU clusters, becomes impedimental. However, it is worthwhile to thoroughly look at different particle modeling techniques because it is only a question of time that soil simulation using particle methods in real-time becomes feasible. That is why, in this chapter, we selected three fundamentally different approaches to model particles in the context of soil simulation. First, we focus on Position Based Dynamics (PBD), a real-time capable model from the Computer Graphics community, see Section 3.2. While being efficient, the computation of forces is cumbersome, and inaccurate. Thereafter, we briefly review Nonsmooth Contact Dynamics (NSCD) in light of the work of [Kle15b], see Section 3.3. Force computations are possible, but the method is so far not real-time capable for relevant number of particles to the author's knowledge. Finally, we look at the classical Discrete Element Method (DEM) and lay special focus on a linear contact model and the nonlinear Hertz-Mindlin-Deresievicz model, see Section 3.4. The accurate force prediction of elastic materials works also with linear models, see [DD04]. The nonlinear model in full detail is even further from real-time capability.

Interacting particle systems may be seen as systems where all particle trajectories describe the system state. When two particles collide, we may assume that the collision happens instantaneously. For perfectly rigid materials, with infinite stiffness, this corresponds to the exact mathematical solution of such a contact phenomenon, which is known under the term nonsmooth contact mechanics [Kle15b]. This allows the choice of less restrictive smoothness assumptions.

On the other hand, every physical material has a finite Young modulus, so, we can consider every collision as smooth on a sufficiently short time scale. This motivates the use of penalty-based collision algorithms. Smoother functions enable the computation of contact forces, which have to be estimated in case of nonsmooth formulations given the impulse and a typical contact duration. Forces arise naturally in the penalty-based DEM context.

Cundall and Strack introduced the Discrete Element or Distinct Element Method DEM in 1971 in their seminal paper, see [CS79]. The idea is as follows: consider distinct elastic objects, that is disks, spheres, polytopes or rigid bodies and calculate their motion separately to assess a realistic bulk behavior. The main difficulty lies in the resolution of the objects contact between each other and the choice of a suitable integration method. There is a rich literature basis. The monograph [OSu11] describes the application of DEM in the context of geotechnical engineering and gives a good overview on the most common penalty-based DEM models. We will present two of them in Section 3.4.

We describe particle systems using Newton's second law

$$M\ddot{\boldsymbol{q}} = \boldsymbol{f}, \tag{3.1}$$

where M denotes the generalized mass matrix, $\ddot{\boldsymbol{q}}$ the generelized acceleration vector and $\boldsymbol{f}$ all external forces. Additionally, we need to prescribe non-penetration and friction constraints. The non-penetration condition yiealds an inequality constraint

$$g(\boldsymbol{q}_i, \boldsymbol{q}_j, t) = \|\Pi_{\boldsymbol{x}}(\boldsymbol{q}_i) - \Pi_{\boldsymbol{x}}(\boldsymbol{q}_j)\|_2 - r_{ij} \geq 0$$

between particle i and j. Here, $\Pi_{\boldsymbol{x}}$ stands for the orthogonal projection of $\boldsymbol{q}$ on to its translational components, that is

$$\Pi_{\boldsymbol{x}} : \mathbb{R}^6 \longrightarrow \mathbb{R}^6, \boldsymbol{q} \mapsto (\boldsymbol{x}, \boldsymbol{0})^T.$$

We incorporate friction via a formulation of Coulomb's law, relating normal and tangential force via an inequality and the friction coefficient μ. Traditionally, one distinguishes between sticking friction μ_s and dynamic friction μ_d, but we simplify Coulomb friction setting $\mu = \mu_s = \mu_d$, as the effect is negligible for our applications.

Consider N particles in motion with indices $i \in \{1, \ldots, N\}$. Considering only one particle, we can rewrite Equation (3.1) for each particle in the form

$$m_i \ddot{\boldsymbol{x}}_i = \boldsymbol{F}_i.$$

Here, m_i denotes the mass of the i-th particle, $\boldsymbol{x}_i$ its position, $\dot{\boldsymbol{x}}_i$ its velocity and $\ddot{\boldsymbol{x}}_i$ its acceleration vector and $\boldsymbol{F}_i$ the sum of the forces acting upon the i-th particle.

Collision Detection Different algorithms exist to search for upcoming collisions. The brute force approach is to check every pair of particles (i, j) for their distance, calculate the overlap $\delta = r_{ij} - \|x_i - x_j\|$ and if it is non-negative, we have detected a collision. This has complexity $\mathcal{O}(n^2)$, so, with a rising number of particles, we run into difficulties.

A second approach, which works especially well for two dimensional discrete element codes, is the use of Delaunay triangulation [Del34; LS80]. Here, we connect the center of mass points, so that they form a collection of triangles. Delauney triangulations guarantee to maximize the minimal angle within a triangle for all triangles. This reduces the problem to checking if a triangle edge is smaller than the maximum particle diameter. Then it suffices to calculate the overlap only for these particles.

For larger particle numbers, it is more efficient to separate the total volume into smaller boxes, using axis-aligned bounding boxes, see [Sch99] with a complexity of $\mathcal{O}(n)$.

Rotation Rotation in dimension $d = 2$ induces one additional degree of freedom, which we denote by ω. If we denote the vertical axis by z and the horizontal axis by x, we rotate around the y-axis. For dimension $d = 3$, we obtain three additional degrees of freedom, corresponding to the rotation of the particle around the x, the y and the z axis. Similarly to the Newton Equation (3.1), the Euler equation

$$\boldsymbol{I}_i \dot{\omega}_i + \omega_i \times (\boldsymbol{I}_i \omega_i) = \boldsymbol{T}_i \tag{3.2}$$

describes the rotation. In case of spheres and disks, rotation symmetry leads to no contribution by the gray part of equation (3.2).

Frame of Reference for a Contact Problem If two discs or spheres i and j come into contact, we obtain a common contact point, which will be the origin of

the contact reference frame. We have a normal direction, whereon we can define two normal unit vectors

$$\boldsymbol{n}_{ij} = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2} \quad \text{and} \quad \boldsymbol{n}_{ji} = -\boldsymbol{n}_{ij} = \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2},$$

where $\boldsymbol{x}_i$ denotes the center of mass of the i-th body.

3.2 Position Based Dynamics

The basic idea is as follows. First we integrate, then we resolve the constraints by adjusting the particle positions. Thereafter, we set the velocities to match the position change.

Rather than solving the particle contact on the acceleration level or on the velocity level, we can directly change the position so that we do not have interpenetration. Müller et al. introduced this approach and coined it PBD in their paper [Mül+06]. It is used in the computer graphics community and within simulations where the contact resolution has little importance. The idea is as follows. If we detect an overlap of two particles, we reset the positions so that the violation is no longer present. Let us describe the procedure of PBD for one particle-particle non-penetration constraint. First, we compute a preliminary position update by

$$\boldsymbol{x}^\star(t_{k+1}) = \boldsymbol{x}(t_k) + \boldsymbol{v}(t_k)\Delta t + (\boldsymbol{g}_z - \gamma \boldsymbol{v}(t_k))\Delta t^2. \tag{3.3}$$

Here $\gamma \boldsymbol{v}_k$ describes a damping term and $\boldsymbol{g}_z$ the gravitation term acting in negative vertical z-direction. Thereafter, we check for a constraint violation by

$$\delta = r_i + r_j - \|\boldsymbol{x}_i^\star - \boldsymbol{x}_j^\star\|_2 = -g_{ij} \geq 0.$$

We further compute the constraint derivatives

$$\boldsymbol{G}_{ij} = \frac{\boldsymbol{x}_i^\star - \boldsymbol{x}_j^\star}{\|\boldsymbol{x}_i^\star - \boldsymbol{x}_j^\star\|_2} = \frac{1}{\delta - r_i - r_j}(\boldsymbol{x}_i^\star - \boldsymbol{x}_j^\star),$$

$$\boldsymbol{G}_{ji} = \frac{\boldsymbol{x}_j^\star - \boldsymbol{x}_i^\star}{\|\boldsymbol{x}_j^\star - \boldsymbol{x}_i^\star\|_2} = \frac{1}{\delta - r_j - r_i}(\boldsymbol{x}_j^\star - \boldsymbol{x}_i^\star).$$

With the compliance $\alpha = \frac{1}{\Delta t^2 E}$, we compute the Lagrange multiplier $\Delta\lambda$ such that $g(\boldsymbol{x} + \Delta\lambda m_i^{-1} G_{ij}) = 0$ by

$$\Delta\lambda = \frac{-\delta}{\boldsymbol{G}_{ij}^T m_i^{-1} \boldsymbol{G}_{ij} + \boldsymbol{G}_{ji}^T m_j^{-1} \boldsymbol{G}_{ji} + \alpha}.$$

Afterwards, we calculate the position update

$$\Delta \boldsymbol{x}_i = m_i^{-1} \boldsymbol{G}_{ij} \Delta\lambda, \quad \Delta \boldsymbol{x}_j = m_j^{-1} \boldsymbol{G}_{ji} \Delta\lambda.$$

We then update the positions $\boldsymbol{x}(t_{k+1}) = \boldsymbol{x}^\star(t_{k+1}) + \Delta \boldsymbol{x}_i$ and set the velocities $\boldsymbol{v}(t_{k+1}) = \frac{\boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k)}{\Delta t}$ accordingly.

Algorithm 1 Position Based Dynamics

for all particles **do**
  $\boldsymbol{x}^*_{k+1} = \boldsymbol{x}_k + \Delta t \boldsymbol{v}_k + \Delta t^2 (\boldsymbol{f}_{ext} - \gamma \boldsymbol{v}_k)$
end for
for all constraints g_ℓ **do**
  $\boldsymbol{x}^*_{k+1} \leftarrow \text{constraintProjection}(\boldsymbol{x}^*_{k+1})$
end for
for all particles **do**
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}^*_{k+1}$
  $\boldsymbol{v}_{k+1} = (\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)/\Delta t$
end for

Friction To incorporate friction, we add an additional tangential constraint [Mac+19, Section 4.4, p. 5]

$$g(\boldsymbol{x}_i, \boldsymbol{x}_j) = D^T (\boldsymbol{x}_i - \boldsymbol{x}_j) = 0,$$

where D is a frictional matrix. After solving for the frictional Lagrange multiplier λ_f, Coulomb friction can be incorporated by

$$\lambda_f' = \min(\mu \Delta\lambda, \lambda_f).$$

Smallsteps The idea is to subdivide each iteration into small substeps, which yield much better results than increasing the number of iterations [Mac+19].

The NVIDIA FleX library implements PBD, which is part of the game engine Unity, see Figure 3.1 and [Haa14]. A slight adaption of PBD has also been successfully applied in the context of soil simulation, see [Hol14]. The question on suitable function spaces for positions and velocities is difficult to answer. As we have seen in Equation (3.3), the preliminary position is updated thus small jumps occur. Whence, we may postulate that $\boldsymbol{x}(t)$ and $\in BV^{3N}([t_0, T], \mathbb{R})$ for $i \in 1, 2, 3$. Similarly the velocities are set after each update to fit the jump in the position. That is, we

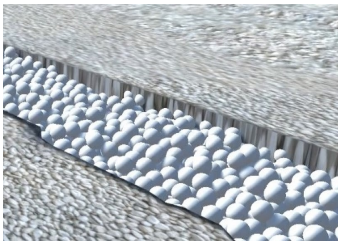


Figure 3.1: Visualization of the plate in trench experiment modelled with Nvidia FLEX in Unity

also have $v_i \in BV^{3N}([t_0, T], \mathbb{R})$. This is – from a mathematically point of view – inconsistent.

That being said, we claim that Position Based Dynamics is an inconsistent particle model. Positions are reset in order to avoid constraint violations, thus positions are already nonsmooth. Velocities are then set accordingly so that the position jump is feasible in one time step. They too, possess jumps. This behavior can be criticized as being nonphysical. Similar to nonsmooth contact dynamics, accelerations may be formulated as measures. A concise formulation of the dynamics is difficult. A precise calculation of forces is, even more than for nonsmooth methods, unclear in most cases.

3.3 Nonsmooth Contact Dynamics

Here, we resolve contact by computing reaction impulses and correct velocities. Afterwards, we update the computed velocities in an Euler step by integrating external forces. We then calculate the new positions.

Instead of solving Newton's second law on the level of accelerations, it is also possible to consider velocities and impulses. This is done in the nonsmooth theory on granular media simulation [Kle15b; TA11]. The idea is to impose Coulomb friction using conical constraints. The impulse γ must be in the local friction cone, whereas the tangential velocity has to be orthogonal to it. The friction coefficient μ defines the opening angle ϕ via the equation $\mu = \arctan(\phi)$.

Motivation A common approach for fast solution of linear equations $A\boldsymbol{x} = \boldsymbol{r}$ is to apply iIterative schemes. Two examples for such schemes are the Gauss-Jacobi and Gauss-Seidel methods, splitting the matrix A into a lower triagonal, a diagonal

and an upper triagonal matrix, i.e. $A = L + D + U$. Given a starting point $\boldsymbol{x}_0$, we obtain an iterative solution by

$$\boldsymbol{x}^{k+1} = D^{-1}(\boldsymbol{r} + (L+U)\boldsymbol{x}^k), \qquad \text{(Gauss-Jacobi)} \tag{3.4}$$

$$\boldsymbol{x}^{k+1} = -(D+L)^{-1}U\boldsymbol{x}^k + (D+L)^{-1}\boldsymbol{r}. \qquad \text{(Gauss-Seidel)} \tag{3.5}$$

This motivates the popular algorithms in the nonsmooth contact dynamics context. There, the contact problem is phrased as a conical constraint. A cone K is a subset of a real vector space $\mathcal{V}$, such that each $\boldsymbol{x} \in K$ can be multiplied with a non-negative scalar $\alpha \geq 0$, and it holds that the product still is in K, i.e. $\alpha\boldsymbol{x} \in K$. A convex cone is a cone that is invariant under non-negative linear combinations, that is $\boldsymbol{x}, \boldsymbol{y} \in K$ and $\alpha, \beta \geq 0$ then $\alpha\boldsymbol{x} + \beta\boldsymbol{y} \in K$. The dual cone is defined by all $\boldsymbol{y} \in \mathcal{V}'$ such that the scalar product with an element of the cone is greater or equal zero, i.e.

$$K^\star = \{\boldsymbol{x} \in \mathcal{V}' \mid \text{ for all } \boldsymbol{y} \in K, \text{ it holds } \langle \boldsymbol{y}, \boldsymbol{x} \rangle_{\mathcal{V}} \geq 0\}.$$

The polar cone is then defined as the negative dual cone, i.e. $K^\circ = -K^\star$. Coulomb friction translates into a projection onto a convex cone K_μ, see [TA11]. A projection in the sense of linear algebra satisfies $\Pi^2(\boldsymbol{x}) = \Pi(\Pi\boldsymbol{x})) = \Pi(\boldsymbol{x})$. We project the impulse $\boldsymbol{\lambda} = (\lambda_n, \lambda_{t1}, \lambda_{t2})^T = (\lambda_n, \boldsymbol{\lambda}_t^T)^T$ in the local contact frame as follows

$$\Pi_{K_\mu}(\boldsymbol{\lambda}) = \begin{cases} \boldsymbol{\lambda} & \text{if } \|\boldsymbol{\lambda}_t\| < \mu\lambda_n \\ \boldsymbol{0} & \text{if } \|\boldsymbol{\lambda}_t\| < -\frac{1}{\mu}\lambda_n \\ \begin{pmatrix} 1 \\ \operatorname{sgn}(\lambda_{t1})\mu \\ \operatorname{sgn}(\lambda_{t2})\mu \end{pmatrix} \frac{\sqrt{\frac{1}{2}(\lambda_{t1}^2+\lambda_{t2}^2)}\mu+\lambda_n}{\mu^2+1} & \text{if } -\frac{1}{\mu}\lambda_n \leq \|\boldsymbol{\lambda}_t\| \text{ or if } \mu\lambda_n \leq \|\boldsymbol{\lambda}_t\|. \end{cases}$$

Impulses in the cone K_μ remain unscathed. If λ is inside the polar cone K°, it is mapped to zero. Otherwise, we project onto the nearest point on the cone K_μ. Indeed $\Pi_{K_\mu}(\Pi_{K_\mu}(\boldsymbol{\lambda})) = \Pi_{K_\mu}(\boldsymbol{\lambda})$. The formulations of the system of equations as a Cone Complementarity Problem reads

$$K \ni N\boldsymbol{\gamma} + \boldsymbol{r} \perp \boldsymbol{\gamma} \in K^\star. \tag{3.6}$$

Here N denotes the product of the constraint derivatives and the mass matrix $N = GM^{-1}G^T$. The right-hand-side vector $\boldsymbol{r}$ collects the velocities induced by the overlap, external forces, and previous velocities. We refer to [Kle15b, Chapter 4.3, p. 87] for a more thorough and detailed discussion.

3.3.1 Projected Gauss-Jacobi Method

Starting from Equation (3.6), we can formulate the following iterative scheme

$$\begin{aligned} \boldsymbol{\gamma}^{r+1} &= \Pi_{K_\mu}(\gamma^r - wD^{-1}(N\boldsymbol{\gamma}^r + \boldsymbol{r})), \\ &= \Pi_{K_\mu}((I - wD^{-1}N)\boldsymbol{\gamma}^r - \omega D^{-1}\boldsymbol{r})), \end{aligned} \tag{3.7}$$

where D denotes a matrix with the diagonal entries of N and w is a relaxation parameter, following [Kle15b, p. 91]. Apart from the projection Π_{K_μ}, this resembles Equation (3.4). The Projected Gauss-Jacobi (PGJ) algorithm is suitable for parallelization, because the loop over non-penetration constraints allows parallelization. Only the sum in the velocity update in line 11 needs sequential evaluation, see Algorithm 2.

Algorithm 2 Projected Gauss-Jacobi

1: **while** iter $< m_I$ **do**
2: **for** all constraints g_{np} **do**
3: $\boldsymbol{\xi_{ij}} = \frac{g_{np}}{\Delta t} + \mu\|\boldsymbol{v}_t\| + \boldsymbol{G}_{ij}\boldsymbol{v}_i + \boldsymbol{G}_{ji}\boldsymbol{v}_j$
4: $\boldsymbol{\delta}_{ij} = \boldsymbol{\lambda}_{ij} - w\eta\boldsymbol{\xi_{ij}}$
5: $\boldsymbol{\delta}_{ij} \leftarrow \Pi_{K_\mu}(\boldsymbol{\delta}_{ij})$
6: $\boldsymbol{\delta}_{ij} \leftarrow \boldsymbol{\delta}_{ij} - \boldsymbol{\lambda}_{ij}$
7: $\boldsymbol{\lambda}_{ij} \leftarrow \boldsymbol{\lambda}_{ij} + \boldsymbol{\delta}_{ij}$
8: **end for**
9: $\boldsymbol{v} = \boldsymbol{v}_0$
10: **for** all particles **do**
11: $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \sum_{j=0}^{N_c} m_{red}^{-1}\boldsymbol{G}_{ij}\boldsymbol{\delta}_{ij} \qquad \boldsymbol{v}_j \leftarrow \boldsymbol{v}_j + m_{red}^{-1}\boldsymbol{G}_{ji}\boldsymbol{\delta}_{ij}$
12: **end for**
13: **end while**
14: **for** all particles **do**
15: $\boldsymbol{v}_{k+1} \leftarrow \boldsymbol{v}_k + g\Delta t$
16: $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \boldsymbol{v}_{k+1}\Delta t$
17: **end for**

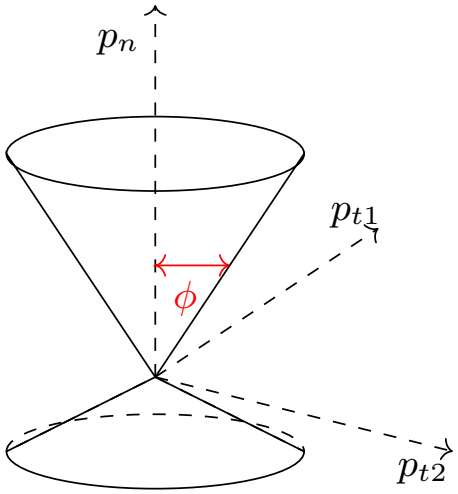


Figure 3.2: Friction cone and projection

3.3.2 Projected Gauss-Seidel Method

The Projected Gauss-Seidel (PGS) solver is less suitable for parallelization but converges faster. The algorithm is presented in Algortihm 3.

NSCD and Reaction Forces NSCD allows for definition of reaction force by reaction impulse as

$$F = G(\boldsymbol{q}_j)\frac{\gamma_j}{\sigma dt + (1-\sigma)dt_c}$$

see [Kle15b, p. 69]. Here, γ_j is the reaction impulse, dt_c is the typical critical time step for a contact event, and σ is an additional meta parameter. Kleinert also describes a concise formulation of the dynamics. We may assume, that the positions $\boldsymbol{x}(t)$ are absolutely continuous. The velocities $\boldsymbol{v}(t)$ are of bounded variation, i.e. $\boldsymbol{v}(t) \in BV^N([t_0, T], \mathbb{R})$. The accelerations exist as a Lebesgue-Stieltjes measure [Kle15b, Chapter 3.2, p. 46].

NSCD and Real-time Applications Renouf et al. applied nonsmooth methods in real-time applications, see [RAD05], but succeeded only for about 160 spheres. Servin et al. use nonsmooth methods in their mutli terrain server, but rely on voxel-based methods for real-time simulation [SBN21].

Algorithm 3 Projected Gauss-Seidel

while iter $< m_I$ **do**
  for all constraints g_{np} **do**
    $\boldsymbol{\xi_{ij}} = \frac{g_{np}}{\Delta t} + \mu\|\boldsymbol{v}_t\| + \boldsymbol{G}_{ij}\boldsymbol{v}_i + \boldsymbol{G}_{ji}\boldsymbol{v}_j$
    $\boldsymbol{\delta}_{ij} = \boldsymbol{\lambda}_{ij} - w\eta\boldsymbol{\xi_{ij}}$
    $\boldsymbol{\delta}_{ij} \leftarrow \Pi_{cone}(\boldsymbol{\delta}_{ij})$
    $\boldsymbol{\delta}_{ij} \leftarrow \boldsymbol{\delta}_{ij} - \boldsymbol{\lambda}_{ij}$
    $\boldsymbol{\lambda}_{ij} \leftarrow \boldsymbol{\lambda}_{ij} + \boldsymbol{\delta}_{ij}$
    $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + m_{red}^{-1}\boldsymbol{G}_{ij}\boldsymbol{\delta}_{ij}, \qquad \boldsymbol{v}_j \leftarrow \boldsymbol{v}_j + m_{red}^{-1}\boldsymbol{G}_{ji}\boldsymbol{\delta}_{ij}$
  end for
end while
for all particles **do**
  $\boldsymbol{v}_{k+1} \leftarrow \boldsymbol{v}_k + g\Delta t$
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \boldsymbol{v}_{k+1}\Delta t$
end for

3.4 Penalty-based Discrete Element Method

Penalty-based DEM is also known under the name soft-sphere approach [Gup15, Chapter 2, p. 27]. Particles can slightly overlap, while this overlap leads to a normal non-penetration force. This results in contrast to the previous models in a system of stiff ODEs.

Formulation as an ODE of First Order First, we reformulate Newton's equation (3.1) in terms of

$$\begin{aligned} M\ddot{\boldsymbol{q}} &= \boldsymbol{F}, \\ \dot{\boldsymbol{q}} &= \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t}. \end{aligned} \tag{3.8}$$

Here, M denotes a block matrix consisting of $I_d * m_i$ and of the moment of inertia $\mathcal{I}_i$, which for spheres with reference point in the center, can also be considered a diagonal matrix. Thus M is a diagonal matrix and has size $6N \times 6N$ or $3N \times 3N$. Let us rewrite equation (3.8) as a first order system by setting $\boldsymbol{u}_1 = \boldsymbol{q}$ and $\boldsymbol{u}_2 = \dot{\boldsymbol{q}}$, thus $\boldsymbol{u} = (\boldsymbol{u}_1, \boldsymbol{u}_2)^T$. Then

$$\dot{\boldsymbol{u}} = (\dot{\boldsymbol{u}}_1, \dot{\boldsymbol{u}}_2)^T = (\boldsymbol{u}_2, M^{-1}\boldsymbol{F}(\boldsymbol{u}_1(t), \boldsymbol{u}_2(t), t)) =: \tilde{\boldsymbol{F}}(\boldsymbol{u}(t), t).$$

Thus, we obtain

$$\dot{\boldsymbol{u}} = \tilde{\boldsymbol{F}}(\boldsymbol{u}(t), t).$$

If $\tilde{\boldsymbol{F}}(\boldsymbol{u}(t), t)$ is continuous in t and Lipschitz continuous in the first argument in the sense that

$$\|\tilde{\boldsymbol{F}}(\boldsymbol{u}(t), t) - \tilde{\boldsymbol{F}}(\boldsymbol{v}(t), t)\| \leq L\|\boldsymbol{u}(t) - \boldsymbol{v}(t)\| \text{ for all } \boldsymbol{u}, \boldsymbol{v} \text{ and } t,$$

we obtain existence and uniqueness of a solution $\boldsymbol{u}$, at least locally, by the Picard-Lindelöf theorem, see Section 2.2.

Time Stepping In penalty-based DEM usually use an explicit Euler scheme, see Section 2.2. This has the advantage of being simple to program. The varying number of contacts per particle is not an issue. However, if the material is stiff, that is the Young's modulus E is large, the time steps need to be small in order to achieve convergence. For such a stiff ODE, one would usually choose an implicit scheme, the simplest being the implicit Euler scheme, see Section 2.2. However, the change in the number of contacts and the requirement of small time steps due to nonlinearity show, that in DEM the disadvantages of the implicit Euler outweigh its advantages, [OSu11, Chapter 2.5, p. 55].

Scale Invariance The terminology of scale-invariant contact laws is due to Feng, see [Fen+09], and applies to all penalty-based interaction laws. The idea is to change the size of the simulated particles, e.g. enlarging particles reduces the total number of particles and larger time steps become stable. This is only plausible, if the relation between stress σ and strain ε is invariant.

This leads to a mesoscopic model where one simulated particle does not correspond to exactly one physical grain. Let us consider physical grains with radius r_P and larger model particles with radius r_M.

Definition 3.4.1. We say that a model is scale-invariant, if and only if for radii r_P and r_M it holds $\sigma_P = \sigma_M$ for $\epsilon_P = \epsilon_M$.

That means, if we enlarge the particles, the strain stress response of the bulk material does not change. This leads to the following theorem.

Theorem 3.4.2. An n-dimensional contact law of the form $F = cr^\alpha \delta^\beta$ is scale-invariant if and only if $\alpha + \beta = n - 1$.

Proof. This result is shown in [Fen+09]. □

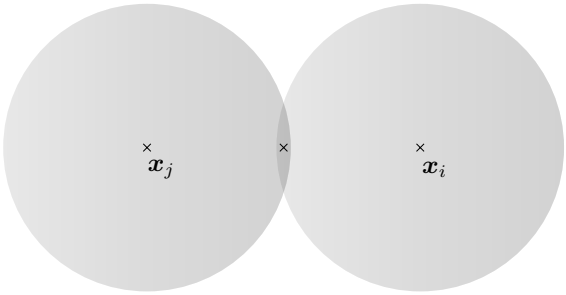


Figure 3.3: Initial contact setup: contact points and actuation point coincide

Initial Contact In the following, we describe a linear DEM model presented in [Obe13]. Two particles with centers $\boldsymbol{x}_i, \boldsymbol{x}_j$, radii r_i, r_j and velocities $\boldsymbol{v}_i, \boldsymbol{v}_j$ collide. To detect a collision, we look at the overlap $\delta_{ij} = r_i + r_j - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$, and when it is non-negative for the first time, we register a collision. The initial contact point $\boldsymbol{x}_{C_{ij}} = \boldsymbol{x}_i + r_i \boldsymbol{n}_{ij}$ ideally coincides with its opposite $\boldsymbol{x}_{C_{ji}} = \boldsymbol{x}_j + r_j \boldsymbol{n}_{ji}$. Furthermore, we set the actuation point as

$$\boldsymbol{x}_{a_{ij}} = \boldsymbol{x}_i + \frac{r_i}{r_i + r_j}(\boldsymbol{x}_j - \boldsymbol{x}_i).$$

In the first time step when a collision is detected, these three points coincide, that is $\boldsymbol{x}_{C_{ij}} = \boldsymbol{x}_{C_{ji}} = \boldsymbol{x}_{a_{ij}}$.

Contact Point in Global and Local Coordinates The initial contact point $\boldsymbol{x}_{C_{ij}}$ is given in a global coordinate frame. We additionally need to save it in local body coordinates

$$\boldsymbol{x}_{C_{ij}}^{loc} = \boldsymbol{x}_{C_{ij}} - \boldsymbol{x}_i$$

and

$$\boldsymbol{x}_{C_{ji}}^{loc} = \boldsymbol{x}_{C_{ji}} - \boldsymbol{x}_j.$$

The global contact point is recomputed in each time step from its local form.

Contact Point in Successive Time Steps If the particle $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in contact, we have to recompute the global contact points from the local contact point. This is done by taking into account any rotations incorporated in the rotation matrix $\boldsymbol{R}_{ij}$, that is $\boldsymbol{x}_{C_{ij}} = \boldsymbol{R}_{ij}\boldsymbol{x}_{C_{ij}} + \boldsymbol{x}_i$. The actuation point is recomputed as the mid-point in the contact area $\boldsymbol{x}_{a_{ij}} = \boldsymbol{x}_i + \frac{r_i}{r_i + r_j}(\boldsymbol{x}_j - \boldsymbol{x}_i)$.

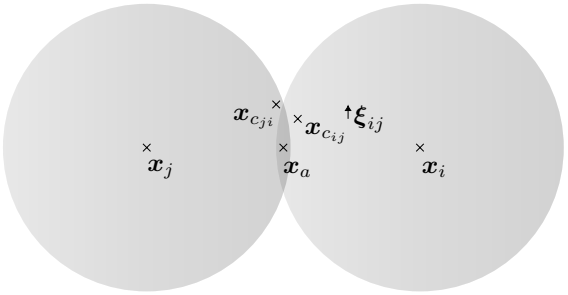


Figure 3.4: By relative translation and rotation, the tangential displacement vector and the rotational displacement vector arise

3.4.1 Hertz-Mindlin-Deresiewicz Model

A nonlinear contact model is based on Hertzian theory and the tangential model by Mindlin and Deresiewicz, [DD04], [Her82], [DM53]. The following discussion is based upon the master thesis [Gho19] and the monograph [OSu11].

Hertzian contact is a scale invariant model with constants $\alpha = \frac{1}{2}$ and $\beta = \frac{3}{2}$, see Theorem 3.4.2. The normal force term reads

$$F_n = \frac{4}{3} E^\star \sqrt{R^\star} \delta_n^{\frac{3}{2}}.$$

Here $E^\star$ is the harmonic mean of Young modulus divided by one minus the Poisson ratio ν of the materials in contact, i.e.

$$\frac{1}{E^\star} = \frac{1-\nu_1}{E_1} + \frac{1-\nu_2}{E_2}.$$

Similarly, $R^\star$ is the equivalent radius

$$\frac{1}{R^\star} = \frac{1}{R_1} + \frac{1}{R_2}.$$

The tangential force is computed by integrating over the contact area with radius a. The sticking area has a smaller radius b. We thus distinguish

$$f_t(r) = \frac{3\mu F_n}{2\pi a^3} \sqrt{a^2 - r^2} \qquad \text{for } b \le r \le a,$$
$$f_t(r) = \frac{3\mu F_n}{2\pi a^3} \left(\sqrt{a^2 - r^2} - \sqrt{b^2 - r^2} \right) \qquad \text{for } 0 \le r < b.$$

The tangential force is then given by

$$F_t = 2\pi \int_0^a f_t(r) r dt = \mu F_n \left(1 - \frac{b^3}{a^3}\right).$$

An even more complex formulation involving a counter-slip region is presented in [OSu11].

3.4.2 Linear DEM Model

A simplified version of the nonlinear Hertz-Mindlin model is a linear contact model. In [DD04], different linear and nonlinear contact models are studied. Their findings indicate, that for granular bulk materials, it suffices to consider linear interaction laws. The model presented here, is based upon the work by [SSE10; Obe+11] and [Fle09, Chapter 2.2.2, p. 11].

Normal Interaction We set $\dot{\delta}_{ij} = \langle \boldsymbol{v}_i - \boldsymbol{v}_j, \boldsymbol{n}_{ij} \rangle$, where $\langle \cdot, \cdot \rangle$ corresponds to the scalar product in $\mathbb{R}^d$. This leads to the computation of a normal conservative force

$$F_{ij}^{N,cons} = k_{ij}^N \delta_{ij}$$

and a normal dissipative force

$$F_{ij}^{N,diss} = d_{ij}^N \dot{\delta}_{ij}.$$

Considering two particles as a stiff beam, with mean radius $r_{ij} = \frac{1}{2}(r_i + r_j)$ and mean area $A_{ij} = \pi r_{ij}^2$. The length of the beam corresponds to $L_{ij} = 2r_{ij}$. Considering the normal stress σ and strain ε

$$\sigma = \frac{F_{ij}^{N,cons}}{A_{ij}} = \frac{k_{ij}^N \delta_{ij}}{\pi r_{ij}^2} \qquad \text{and} \qquad \varepsilon = \frac{\delta_{ij}}{2r_{ij}},$$

we obtain for the Young modulus

$$E_N = \frac{\sigma}{\varepsilon} = \frac{k_{ij}^N 2r_{ij}}{\pi r_{ij}^2} \qquad \text{thus} \qquad k_{ij}^N = \frac{E_N \pi r_{ij}}{2}$$

Hence, the presented model, neglecting the damping term, describing a 3-dimensional contact law with $n = 3$ and $\alpha = \beta = 1$, is scale-invariant. We set the inter-particle damping

$$d_{ij}^N = D_N 2\sqrt{k_{ij}^N m_{ij}}, \qquad \text{where} \qquad m_{ij} = \frac{m_i m_j}{m_i + m_j}.$$

The effective mass m_{ij} stems from the consideration of two particles as one damped oscillator. The parameter D_N controls the desired percentage of critical damping. In summary, we obtain

$$\boldsymbol{F}_{ij}^N = \left(k_{ij}^N \delta_{ij} + d_{ij}^N \dot{\delta}_{ij}\right) \boldsymbol{n}_{ij}.$$

Tangential Force By translation or rotation of the two particles with respect to each other, the contact point moves with the particles center of gravity. Thus, the global contact points may not coincide, see Figure 2. The tangential displacement is the projection of the vector connecting the contact points into the tangential plane. By transformation of the local contact points, we obtain the global contact points

$$\boldsymbol{x}_{C_{ij}} = I\boldsymbol{x}_{C_{ij}}^{loc} + \boldsymbol{x}_i.$$

The identity matrix I has to be replaced by the respective rotation matrix, if rotations are to be considered. Now, let us consider the contact point displacement

$$\boldsymbol{\xi}_{ij}' = \boldsymbol{x}_{C_{ji}} - \boldsymbol{x}_{C_{ij}}$$

and after projection into the tangential plane the tangential displacement

$$\boldsymbol{\xi}_{ij} = \Pi\boldsymbol{\xi}_{ij}' = \boldsymbol{\xi}_{ij}' - \langle \boldsymbol{\xi}_{ij}', \boldsymbol{n}_{ij}\rangle \boldsymbol{n}_{ij}.$$

The relative velocity $\boldsymbol{v}_i - \boldsymbol{v}_j$ is projected into the tangential plane, more specifically we set

$$\dot{\boldsymbol{\xi}}_{ij} = \langle \boldsymbol{v}_i - \boldsymbol{v}_j, \boldsymbol{t}\rangle \boldsymbol{t},$$

where $\boldsymbol{t} = \frac{\boldsymbol{\xi}_{ij}}{\|\boldsymbol{\xi}_{ij}\|}$ is the tangential unit vector. We consider a tangential conservative force

$$\boldsymbol{F}_{ij}^{T,cons} = -k_{ij}^T \boldsymbol{\xi}_{ij}$$

and a tangential dissipative force

$$\boldsymbol{F}_{ij}^{T,diss} = -d_{ij}^T \dot{\boldsymbol{\xi}}_{ij}.$$

Together, we obtain $\boldsymbol{F}_{ij}^T = -k_{ij}^T \boldsymbol{\xi}_{ij} - d_{ij}^T \dot{\boldsymbol{\xi}}_{ij}$.

Coulomb friction If the tangential force is high, with respect to the normal force, slipping occurs. This is accounted for, by reducing the tangential force, acting in opposite direction to the particle velocity. More specifically, we enforce

$$F_{ij}^{T,cons} \leq \mu F_{ij}^N. \tag{3.9}$$

This is done, before calculating the dissipative part of the tangential force. If the Coulomb friction condition (3.9) is violated, the tangential force is reset to

$$\boldsymbol{F}_{ij}^T = \mu F_{ij}^N \frac{\xi_{ij}}{\|\xi_{ij}\|}.$$

Rotation Rotation arises due to a torque induced by a tangential force acting at the actuation point, that is

$$\boldsymbol{T}_i = (\boldsymbol{x}_{a_{ij}} - \boldsymbol{x}_i) \times \boldsymbol{F}_T.$$

The rotation needs to be taken into account when transforming the contact point from local to global coordinates. We can further introduce rolling resistance to model irregular shapes that inhibit excessive rolling and allow steep angles of repose, see [Obe13, Chapter 4.2.3, p. 30].

The whole model is shortly summarized in Algorithm 4.

Symbol	Name	Range	Unit
Linear normal model			
r_i	radius	$0.0001 - 0.1$	m
n	porosity	$0.3 - 0.6$	–
ρ	density	$1000 - 3500$	$\mathrm{kg/m^3}$
$\tilde{m}_{ij}$	reduced mass	$\frac{m_i m_j}{m_i + m_j}$	kg
E	Young modulus	$10^5 - 10^8$	$\mathrm{N/m^2}$
D_N	damping coefficient	$[0 - 1]$	–
k_{ij}^N	normal stiffness	$\frac{\pi}{2}\frac{r_i + r_j}{2}E$	N/m
d_{ij}^N	normal damping	$D_N 2\sqrt{\tilde{m}_{ij} k_{ij}^N}$	Ns/m
Linear tangential model			
k_{ij}^T	tangential stiffness	$\frac{1}{1.2}k_{ij}^N$	N/m
D_T	damping coefficient	$[0 - 1]$	–
d_{ij}^T	tangential damping	$D_T 2\sqrt{\tilde{m}_{ij} k_{ij}^N}$	Ns/m
Coulomb friction			
μ	friction coefficient	$[0 - 1]$	–

Table 3.1: Parameters of the DEM model, with range of the parameters and physical unit.

3.5 Comparison of Different Models

The three methods from this chapter all have their pros and cons. The classical DEM resolves contact with a smoothing penalty term on the acceleration level. This requires small time steps but is suitable for direct computation of particle-tool forces. Classical ODE theory yields convergence of the method.

Algorithm 4 DEM - Linear Contact Model

procedure ContactParticleParticle(Contact,x_i, x_j) ▷ Particles i and j are in Potential Contact
  if $\delta_{ij} = r_i + r_j - \|x_i - x_j\| > 0$ **then**
    if Contact (i, j) not Initialized **then**
      Initialize Contact (i, j)
      $\boldsymbol{x}_{C_{ij}} = \boldsymbol{x}_{C_{ji}} = \boldsymbol{x}_{a_{ij}} = \boldsymbol{x}_i + \frac{r_i}{r_i+r_j}(\boldsymbol{x}_i - \boldsymbol{x}_j)$
      $\boldsymbol{x}^{loc}_{C_{ij}} = \boldsymbol{x}_{C_{ij}} - \boldsymbol{x}_i$
      $\boldsymbol{x}^{loc}_{C_{ji}} = \boldsymbol{x}_{C_{ji}} - \boldsymbol{x}_j$
    else
      $\boldsymbol{x}_{a_{ij}} = \boldsymbol{x}_i + \frac{r_i}{r_i+r_j}(\boldsymbol{x}_i - \boldsymbol{x}_j)$
    end if ▷ Normal Contact
    $\dot{\delta}_{ij} = \langle \boldsymbol{v}_i - \boldsymbol{v}_j, \boldsymbol{n}_{ij} \rangle$
    $F^N_{ij} = k^N_{ij}\delta_{ij} + d^N_{ij}\dot{\delta}_{ij}$
    ▷ Tangential Contact
    $\boldsymbol{x}_{C_{ij}} = \boldsymbol{R}_{ij}\boldsymbol{x}^{loc}_{C_{ij}} + \boldsymbol{x}_i$
    $\boldsymbol{x}_{C_{ji}} = \boldsymbol{R}_{ji}\boldsymbol{x}^{loc}_{C_{ji}} + \boldsymbol{x}_j$
    $\boldsymbol{\xi}'_{ij} = \boldsymbol{x}_{C_{ji}} - \boldsymbol{x}_{C_{ij}}$
    $\boldsymbol{\xi}_{ij} = \boldsymbol{\xi}'_{ij} - \langle \boldsymbol{\xi}'_{ij}, \boldsymbol{n}_{ij} \rangle \boldsymbol{n}_{ij}$
    $\boldsymbol{F}^{T,cons}_{ij} = k^N_{ij}\boldsymbol{\xi}_{ij}$
    if (**then** $F^{T,cons}_{ij} > \mu F^N_{ij}$) ▷ Coulomb Friction
      $\boldsymbol{F}^T_{ij} = \mu F^N_{ij} \frac{\xi_{ij}}{\|\xi_{ij}\|}$
      $\boldsymbol{\xi}'_{ij} = \mu \frac{F^N_{ij}}{k^T_{ij}} \frac{\xi_{ij}}{\|\xi_{ij}\|}$
      $\boldsymbol{x}'_{C_{ij}} = \boldsymbol{x}_{C_{ij}} - \frac{\boldsymbol{\xi}'_{ij}}{2}$
      $\boldsymbol{x}'_{C_{ji}} = \boldsymbol{x}_{C_{ji}} + \frac{\boldsymbol{\xi}'_{ij}}{2}$
    else ▷ Dissipation
      $\dot{\boldsymbol{\xi}}_{ij} = \langle \boldsymbol{v}_i - \boldsymbol{v}_j, \boldsymbol{t} \rangle \boldsymbol{t}$
      $\boldsymbol{F}^{T,diss}_{ij} = d^T_{ij}\dot{\boldsymbol{\xi}}_{ij}$
    end if
  end if
end procedure

In contrast, the nonsmooth algorithms focus on the velocity level. The jumps in the velocities seem physically accurate for materials with infinite stiffness. The algorithms allow much larger stable time steps. However, the computation of reaction forces is not so straight forward. The typical contact duration has to be taken into account to obtain forces from the computed impulses.

Position Based Dynamics form a relatively new class of algorithms, which in a certain way tends to ignore Newtonian physics. Contact is resolved on the position level, leading to visually pleasing results, but the effects in the velocities may become nonphysical. Visually pleasing simulations can be achieved in realtime, but force computation in a physically accurate way is difficult.

We implemented a prototypical version of the above algorithms for two dimensional problems in MATLAB, see [Mat; Jah]. This served as a pre-study to deepen the author's understanding regarding physical particle models. The code is not optimized but rather kept as general so that it can serve for the study of different algorithms.

In all three scenarios the material stiffness in form of Young's modulus E is important, since it affects the compliance (PBD and NSCD) or the stiffness. Furthermore, the damping parameters D_n (DEM), w (NSCD) and γ (PBD), have an influence on the simulation which is difficult to compare. So although we developed a prototype for disks for most of the presented models, a thorough comparison is still difficult.

Numerical Examples In order to show the differences of the three models, we present two numerical examples. Out of each class of algorithms, we focused only on one for this study, namely on standard PBD, Projected Gauss-Seidel (PGS) as a representative nonsmooth method (on the velocity level) and the linear penalty-based DEM model (on the acceleration level). The relevant parameters for the simulation are shown in Table 3.2, the source code is publicly available at Github [Jah].

First, we will look at the simulation of the bouncing ball example, a famous academic one dimensional example which has been cited in numerous publications in different scenarios [GST12, Example 2.12, p. 25], [SA14, Problem 4.2, p. 192]. A ball with diameter $d = 1$ is positioned at a height of $z = 2$ m under the influence of gravity, see Figure 3.5. This shows the main difference of PBD as being elastic and PGS as a nonsmooth representative as completely rigid (NSCD), in contrast to the bouncing effect of DEM. For this example, we took the linear DEM solution as a reference. For PBD, we varied the damping parameter γ. If we choose $\gamma = 0$, than no damping is in effect and the bouncing will continue indefinitely. For $\gamma = 5$, we still get higher bouncing compared to the DEM. $\gamma = 10$ leads to a similar behavior

Symbol	Name	Value	unit
General parameters			
ρ	density	2700	$\mathrm{kg/m^3}$
E	Young modulus	10^8	$\mathrm{N/m^2}$
μ	friction coefficient	0.3	–
PBD and PGS			
n_{steps}	iterations	1	–
PGS			
w	relaxation factor	0.2	–
PBD			
γ	Damping	8	1/ms
Linear DEM			
D_N	Damping coefficient	0.2	–
D_T	Damping coefficient	0.02	–

Table 3.2: Parameters for the numerical examples for different models. In Bouncing Ball Example the number of steps n_{steps} and the damping γ has been adapted accordingly.

as for DEM, considering the first arc at $t = 1$ second. The PGS solver - if full convergence is desired and the maximum iteration number m_I is set very high - does not bounce at all. By reducing it to $m_I = 10$, we see a small bouncing and for $m_I = 1$ the behavior is close but slightly lower compared to the DEM solution. To be fair, PGS would converge to an inelastic collision with a wall of infinite mass. This could be overcome by increasing the impulse after collision or by introducing some coefficient of restitution, see [SA14, Problem 4.2, p. 192]. Notice that the artificial parameter γ in the PBD solution already influences the trajectory before the first contact has happened, as the green trajectories decay slower than the blue or the red one.

A second example is about a 2 dimensional pile of particles collapsing and forming an angle of repose, see Figure 3.6. In this example, we want to focus on the maximal stable integration time step and the total execution time of our MATLAB code. The code is not optimized for speed, however it gives us a rough idea of how efficient the methods are, compared to each other. We consider a sample of $N = 200$ particles with a radius distribution between 2.5 and 5 cm. We compute 1 second simulation time and measure the computation time, as indicated in Table 3.3. Note that the parameter m_I in PGS was chosen to be 1, so the solver does not fully converge. The computation was performed on a Lenovo T490s with an Intel i7-8665 4 core processor and 16 GB DDR4 Ram. In this example, PBD has a real-time

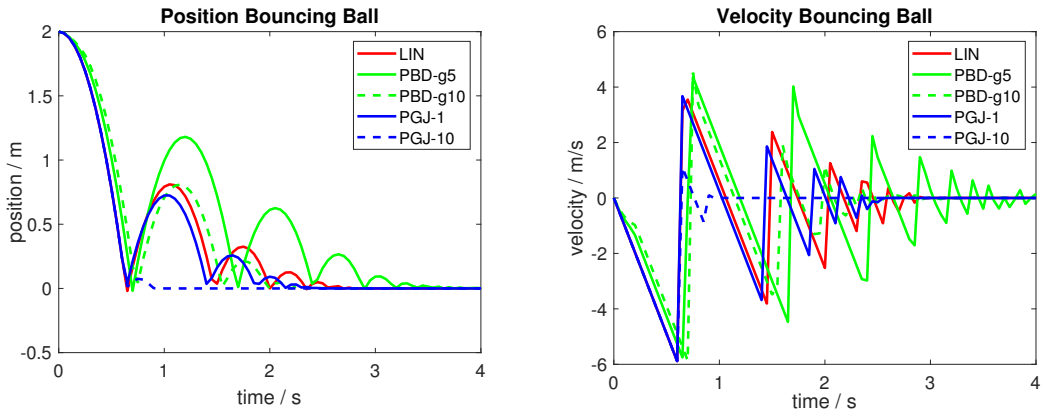


Figure 3.5: Bouncing ball example:Position (Left), Velocity (Right). We compare the linear DEM model (red), with the PBD for $\gamma \in \{5, 10\}$ and PGS for the maximum iteration $m_I \in \{1, 10\}$

factor of 8, while for the linear DEM model, we only achieve a factor of 58. To put this into perspective, GRAPE achieves real-time factors of about 100 for industrial size problems of about 150 thousand three dimensional non-rotational particles run in parallel on a CPU cluster [Bur+17]. And indeed, there exist real-time capable PBD implementations, see Figure 3.1, but which do not allow good force predictions. For larger time steps than 0.0025 seconds, all three methods showed nonphysical behavior. The DEM solver, did not converge for time steps larger than 0.001 second.

Model $\backslash \Delta t /$ s	$5 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$
PBD	42.0	21.0	8.32
NSCD	56.7	26.7	11.3
DEM	123.5	58.5	-

Table 3.3: Efficiency of different modeling approaches. Runtimes for 1 second simulation of collapsing 200 particle pile.

In a third example, we want to study the scaling with the particle size of the different algorithms. Similarly to the second example, we chose three problems, a small one with 100, a medium size with 200 and a large problem with 400 particles. We measure the time to simulate 1 second simulation time of the collapsing pile. In Figure 3.7, we see the total simulation time as a function of the number of particles for linear DEM, the PBD and the PGS algorithm. As a reference, we plot a dashed

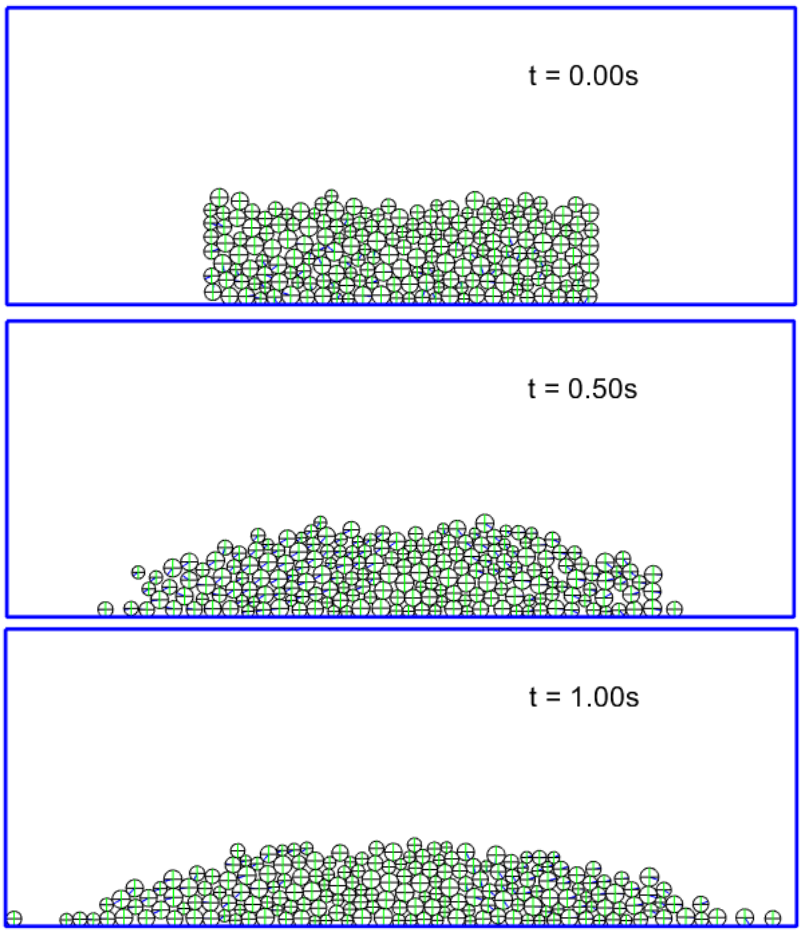


Figure 3.6: Collapsing particle pile consisting of 200 polydisperse particles

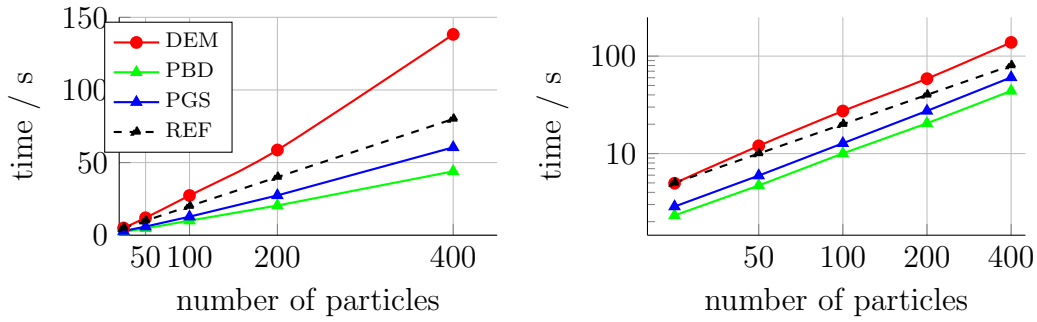


Figure 3.7: All three algorithms scale superlinearly with the number of particles

Model	Force Prediction Quality	Efficiency	Real-time capability
PBD	-	+	+
NSCD	0	0	-
DEM	+	-	-

Table 3.4: Comparison of the different particle methods.

linear reference given by $f(x) = 0.2x$. The total simulation time is shown linearly (left) and as a log–log plot (right), where the linearity is clearly visible.

Discussion At the start of this Chapter, we defined two goals for our comparison. First, the three models are evaluated with respect to their suitability for direct application in real-time, which is covered in detail in Table 3.4. Here, PBD is most efficient and we know, that it can be used in realtime. NSCD is less efficient, but in the future, simple nonsmooth algorithms covering only the main aspects might run in realtime if hardware development continues. Penalty-based DEM has a limitation regarding the stiff nature of the underlying ODE. Thus, the time steps need to be small enough in order to achieve convergence. The second goal is the suitability of the methods for the generation of offline training data. Here, non-rotational DEM results in accurate force predictions, see also Chapter 5. Force prediction for NSCD is possible, but not straight forward, see [Kle15b]. While being very efficient, the PBD formulation does not allow a simple way of tool force computations. Thus for our second goal, we prefer penalty-based DEM, see also Table 3.5.

While forces and torques are at the focus of our studies, we will restrict our self to the linear DEM model, see Section 3.4.2, from now on. Rotations will be neglected as it is known that the effect on tool forces is negligible, see [Obe13]. This model suffices for accurate force prediction, as we also demonstrate in 5.3. PBD has the capability of running in realtime, but lacks the possibility of accurate force predictions. For high numbers of particles, DEM and NSCD are not real-time capable.

3.6 Parametrization of Soil

What we have omitted in this chapter so far, is the correct parametrization of any of the mention models. In academia, the choice and the study of suitable model is often the main focus. Equally important is the ability to calibrate all free parameters of a model in order to have good agreement with a physical system. The

Model	Goal 1: Real-time Application	Goal 2: Offline training data
PBD	+	-
NSCD	0	0
DEM	-	+

Table 3.5: Comparison of the different particle methods regarding direct application in realtime (Goal 1) or for the computation of offline data (Goal 2)

parametrization requires measurement data of a standardized test. Afterwards, a second measured experiment can be simulated leading to the validation of the model. Some parameters, such as masses, length and time scales can be incorporated into the models directly. However, there often remain free parameters in a model which allow the calibration. Often, it is helpful to run a sensitivity analysis of a model with respect to the free parameters. In this section, we will first talk about parametrization in general, referring to all particle models of this chapter. Thereafter, we will present a short digression into soil mechanics and focus on the parametrization of the linear penalty-based DEM model, see Section 3.4.2. This will serve as a foundation for upcoming parametrizations and offline simulations of the data-based models, see Section 5.2 and 4.3.

Each of the three presented particle models in this chapter contains several parameters, which have to be set accordingly in order to obtain results which correspond to physical phenomena. We distinguish three different parameter sets. First, there are obvious correspondences in certain parameters. These comprise parameters as mass m or particle density ρ. Second, there are parameters related to the particle pile, such as the radius distribution r_i or the porosity n of a sample. Third, there are free parameters related to the contact model. These comprise Young's Modulus E related to stiffness, but also damping and friction coefficient μ. In PBD, the damping coefficient γ has to be calibrated. In the presented NSCD algorithms the parameter w and the number of iterations n_{steps} have an impact on the result [KOB13]. In the linear DEM, the damping coefficients D_N and D_T have to be set accordingly.

Uncertainty The generation of a model sample with a certain porosity involves a large portion of randomness. Either we let particles fall into a box under the influence of gravity until they form a stable pile. Or we place particles on a regular grid and shrink this box until a certain porosity is obtained. In both approaches, the formation of patterns in the reference sample is subject to randomized positioning

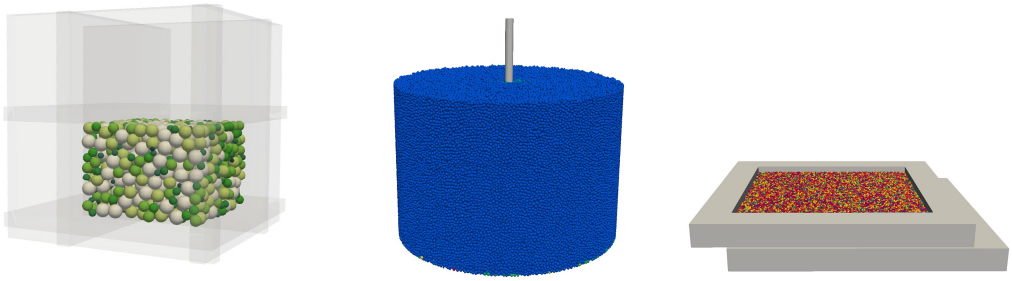


Figure 3.8: Visualization of the different soil mechanical calibration tests. From left to right: Triaxial Test, Penetrometer Test and Direct Shear Test

or collisions between the particles. Indeed, the packing of granular materials is random and one can draw similarities to chaotic systems, see [Gib17, Section 9.3, p. 105].

Soil Mechanical Foundation Throughout this thesis, we distinguish physical grains, i.e. complex shaped physical soil and simulated particles. We are interested in the bulk behavior of many physical grains or simulated particles. The total grain volume V_g as the sum of all soil grain volumes and the void volume V_v form the total bulk volume V_t. The porosity n is then defined as

$$n = \frac{V_v}{V_t} = \frac{V_t - V_g}{V_t} = 1 - \frac{V_g}{V_t} = 1 - \frac{\rho_g}{\rho_b}.$$

Here, the grain density $\rho = \rho_g$ describes the mass of particles per volume. In contrast, ρ_b describes the bulk density of the material. The bulk material corresponds to a continuous description of particles and surrounding void. A soil sample is characterized by sieving its constituents and results in a grain size distribution curve. The mechanical properties of soil are measured in form of a cohesion c and an angle of friction ϕ with different tests. The most common approach is the Triaxial Compression Test (TT), described in more detail in Section 5.2, but we also studied Small Scale Cone Penetration Tests in [Jah+19] and Direct Shear Test in [Ste+21], see Figure 3.8.

Laboratory Measurements In a soil laboratory the triaxial test is performed using different sidewall pressure levels in order to obtain the axial-strain-stress curves. The triaxial test is described in more detail in [Kol07, Chapter 8.5, p. 131]. A cylindrical sample is put into a triaxial cell, we assume dry material, so drainage

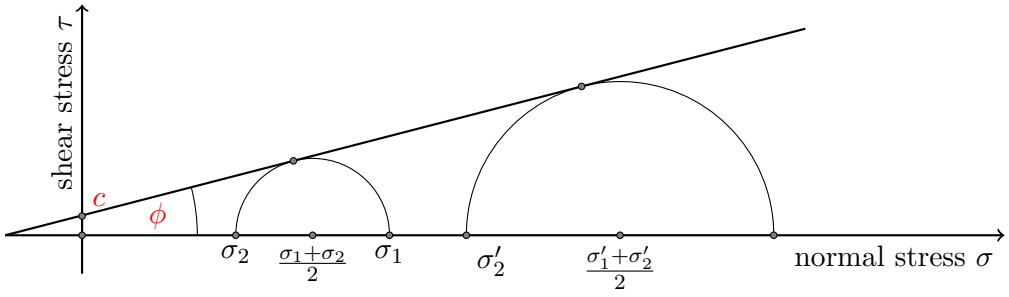


Figure 3.9: Mohr Coulomb Circles

is not required. The sidewall pressure is kept constant, while the axial pressure from vertical z-direction increases. The piston is quasi statically depressed with constant vertical velocity. Loosely packed soil samples are just compacted, the volumetric strain decreases (contraction), compare to the strain-stress curves in the Post Processing tile of Figure 3.10. Densely packed samples initially have a contraction phase, then enter into dilatancy, thus the volume increases. The change from contraction to dilatancy corresponds to the soils failure. This can also be observed in the strain-stress diagram. Here, the axial strain increases until we reach a maximum, the peak stress. When the soil fails, i.e. when a failure zone arises, the stress starts to converge to a residual stress. For loosely packed samples, we do not attain the peak stress, the stress curve converges from below towards the residual stress. For densely packed samples, the soil failure coincides with a stress decrease from peak stress in direction to the residual stress. The relation between axial stress σ_1 and sidewall pressure σ_2 of either peak or residual stress for different sidewall pressures express the shearing behavior. We obtain the friction angle ϕ and the cohesion c by drawing a tangential interpolation on two or more circles, as illustrated in Figure 3.9.

This procedure is usually usually reserved for small grained soil, like sand or silt. However, using larger non-standard triaxial cells, the triaxial test can also be performed for gravel. The volumetric-strain-strain curves is difficult to measure correctly for coarse grained material, like gravel. Moreover, in the simulated samples, the strain-stress relation is more relevant compared to the volumetric behavior. Furthermore, the grain size distribution is measured in the soil laboratory, using different coarse sieves. Large grains like stones or gravel are selected manually. Fine material, like silt has to be wet sifted. We determine the weight of the sieved material and obtain the mass percentage of different intervals of the particle diameters.

The Parametrization Procedure within GRAPE Within GRAPE, we rely upon two steps for the generation of a suitable calibrated particle sample. First, we generate the particle pile, relying upon the measured grain size distribution and the porosity. This step is referred to as sample generation. Second, we focus on the shearing behavior based on the TT calibrating the remaining parameters. The parametrization is mainly based on the following parameters:

- the grain size distribution corresponding to the particle radii r_i
- the material density ρ of the particles
- the porosity n of the soil
- the Young modulus E of the particles
- the normal damping D_N of the particles
- the friction coefficient μ
- the cohesion c

We she the particle density ρ exactly as for the measured material to ensure mass conservation. The particle size distribution, i.e. the radii r_i and the porosity n is defined in the sample generation step. Normal damping is typically set to $D_N = 0.1$ and when considering dry material, the cohesion is set to $c = 0$ Pascal. The remaining free parameters are the Young's modulus E and the friction coefficient μ. Calibrating these two with different TT simulations is the main task of the parametrization process. Both parameters significantly affect the shearing behavior, see [Obe13, Section 5.2, p. 60ff] for a sensitivity study.

Sample Generation The sample generation for a reference volume can be performed as follows. The grain size distribution curve is often upscaled, so that we obtain a mesoscopic particle sample with larger radii. This is possible, due to the property of scale invariance of the contact law, see Paragraph 3.4. That is, the particle size distribution is thus shifted to the right. Particles are then placed on a rectangular grid which is larger than the desired volume and are slightly perturbed in an arbitrary direction. Thereafter, the the particles are compressed, until the desired volume and the desired porosity n is obtained.

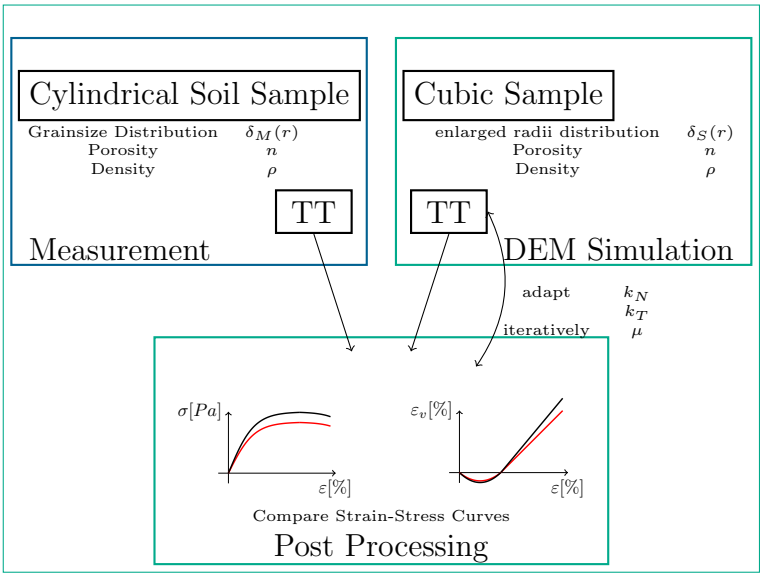


Figure 3.10: Overview of the parametrization procedure with measurements (left), simulation (right) and post processing (below). In the post processing step we compare strain-stress and volumetric strain-strain curves.

Simulation of true Triaxial Tests The triaxial test is simulated with the generated particle sample. Hereby, we make use of a slight simplification. Instead of using a cylinder filled with soil, as described in the DIN18137-1:2010-07, we use a cube with servo-controlled side-walls. This procedure has been used in triaxial tests depending on temperature, see [HA90], and is known under the name true triaxial test. The TT measurements as described above reduce the stress in two dimensions, namely the radial stress corresponding to the side wall pressure and the axial stress. The first three parameters have to be chosen in the sample generation process. The latter parameters can be adjusted while performing a parameter study simulating the TT. The cohesion is only relevant for sticky, wet or very fine materials and can normally be neglected. Thus, as mentioned previously, the relevant parameters in the calibration are Young's Modulus E, friction coefficient μ.

4 Surrogate Models for Online Approximation

In this Chapter, we will focus upon models suitable for online approximation. In the preceding chapter, we presented three different particle models and compared them with respect to efficiency, parameters, force prediction accuracy, and practicability. Regarding force predictions, the classical penalty-based DEM is best suited, but the small integration steps impede the pursuit of real-time capability. That is why in this chapter, we focus on alternative surrogate models, which are more efficient, but less physically intuitive. First, we consider the oldest earthmoving model based on Coulombs work [Cou76]. Thereafter, we consider different prototypical ideas regarding hybrid particle models. Finally, we present the machine learning solutions implemented at the driving simulator, namely a DEM LUT and a DEM RNN approach.

4.1 Fundamental Earthmoving Equation

The Fundamental Earthmoving Formula (FEE) was first derived in [Ree64]. Here, the authors takes up the ideas of Coulomb of studying a two dimensional earth wedge to compute the resisting forces. The following derivations are based upon the book on soil mechanics [McK85], the article by [Sin97] and the master thesis [Can99]. [PPG11] et al. extend the study and prescribe explicit values for the parameters of the model. Bennett et al. applied the FEE in the context of excavation, see [Ben+16].

We start by looking at a soil wedge as depicted in Figure 4.1, and compute the static force equilibrium. We look at a tool depicted in green moving a wedge of soil. The tool angle with respect to the soil surface is denoted by α, the tool length by L_t. We want to compute the static tool force F by looking at it componentwise, that is we consider FX and FZ independently. We protrude the wedge model by a tool width w, in order to define a soil volume V. The soil with density γ and volume V experiences an acceleration due to gravity denoted by g. The soil

will break at a failure surface L_f forming a wedge. The critical failure angle β is given by minimizing the force equilibrium. The soil's friction angle ϕ specifies the direction of the underlying soil-soil reaction force.

Let us first look at the force equilibrium in longitudinal x- and vertical z-direction, compare to Figure 4.1. We decompose F into its components $F\sin(\alpha)$ and $F\cos(\alpha)$.

For the soil resistance force R, we first look at the components $R\sin(\phi)$ parallel and $R\cos(\phi)$ orthogonal to the plane L_f. Using the soil failure angle β, we obtain that the longitudinal components comprise

$$-\sin(\beta)\cos(\phi)R - \cos(\beta)\sin(\phi)R = -\sin(\beta+\phi)R.$$

Similarly, by collecting all terms in vertical z-direction we obtain

$$\cos(\beta)\cos(\phi)R - \sin(\beta)\sin(\phi)R = \cos(\beta+\phi)R.$$

The wedge volume V can be decomposed into two protruded triangles of width w. The weight of the wedge is Therefore, we express the left triangle volume as $\frac{1}{2}\cot(\alpha)d \times d$ and the right one as $\frac{1}{2}\cot(\beta)d \times d$. In total, we obtain $V = \frac{1}{2}d^2w(\cot(\alpha)+\cot(\beta))$. The static force equilibrium yields:

$$\sum F_x = \sin(\alpha)F - \sin(\beta+\phi)R = 0, \tag{4.1}$$

$$\sum F_z = \cos(\alpha)F - \gamma g dw\left(\frac{1}{2}d\left(\cot(\alpha)+\cot(\beta)\right)\right) + \cos(\beta+\phi)R = 0. \tag{4.2}$$

Thereafter, we may resolve the first equation (4.1) to obtain

$$R = \frac{\sin(\alpha)F}{\sin(\beta+\phi)}.$$

Plugging this into the second component equation (4.2), we obtain the simplified earthmoving formula

$$F = \gamma g d^2 w \frac{\cot(\alpha)+\cot(\beta)}{2(\cos(\alpha)+\cot(\beta+\phi)\sin(\alpha))} = \gamma g d^2 w N_\gamma, \tag{4.3}$$

where N_γ collects all trigonometrical terms.

Additionally, cohesion forces F_c depending on the coefficient c along the failure surface and adhesion forces F_{c_a} depending on the coefficient c_a along the tool length may be incorporated. The failure surface can be expressed by the cutting depth

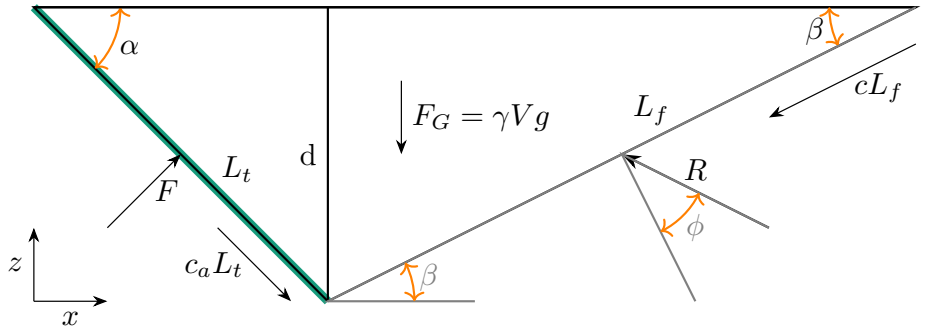


Figure 4.1: Soil wedge for the derivation of the Fundamental Earthmoving Formula including cohesion and adhesion

d by $L_f = d/\sin(\beta)$. The We skip the full derivation here and just mention the formula including the additional terms

$$\begin{aligned} F &= w\gamma g d^2 \frac{1}{2} \frac{(\cot(\alpha) + \cot(\beta))}{(\cos(\alpha) + \cot(\beta+\phi)\sin(\alpha))} \\ &\quad + c_a w d \frac{(1 - \cot(\alpha)\cot(\beta+\phi))}{(\cos(\alpha) + \cot(\beta+\phi)\sin(\alpha))} - cwd \frac{(1 + \cot(\beta)\cot(\beta+\phi))}{(\cos(\alpha) + \cot(\beta+\phi)\sin(\alpha))} \\ &= \left(\gamma g d^2 N_\gamma + c_a d N_{c_a} - c d N_c\right) w. \end{aligned} \tag{4.4}$$

Here, again the N-factors N_{c_a} and N_c collect the trigonometric terms in Equation (4.4). The surcharge material forming in front of the tool, amounts to the additional term $Q = qd(\cot(\alpha) + \cot(\beta))$.

Considering dynamic effects, resulting in the consideration of the velocity of the tool, the total force reads

$$F = (\gamma g d^2 N_\gamma + c_a d N_{c_a} - c d N_c + \gamma g v^2 d N_v + q d N_q) w, \tag{4.5}$$

with $N_q = 2N_\gamma$ and

$$N_v = \frac{\tan(\beta) + \cot(\beta+\phi)}{(\cos(\alpha) + \sin(\alpha)\cot(\beta+\phi))\,(1 + \tan(\beta)\cot(\alpha))}$$

Equation (4.5) can be found in [McK85, Equation (3.68), p. 72]. The cohesion c and adhesion c_a can be neglected.

Computing the Critical Failure Angle β Apart from the soil mechanical parameters of density γ, and friction angle ϕ, the geometric configuration of the tool yields L_t, α and d. What remains undetermined so far is the failure angle β and, directly related to it, the failure surface L_f. Therefore, we need to solve the optimization problem

$$\beta_{crit} = \min_{\beta} F(\beta).$$

This can be understood as finding the angle where the soil failure surface arises under a minimal force input.

Drawbacks of the FEE The main problem with the use of the FEE as presented in Equation (4.3) is its instability. Apart from being a huge simplification of a soil wedge, the denominator may cause problems if it gets close to infinity. This numerical problem happens, when $(\cos(\alpha) + \cot(\beta + \phi)\sin(\alpha)) \to 0$.

4.2 Physical Hybrid Particle Models

The FEE is real-time capable, but as just discussed, unstable and too simple for accurate force prediction. Therefore, we will consider hybrid models which are closely related to the physical particle model and are potentially real-time capable. We have only implemented prototypes of some of the following models.

4.2.1 Adaptive Particle Merging

This approach aims at model order reduction of the DEM in form of a solid mechanics approach. Particles are glued together and treated as one multibody in order to reduce the integration and computation times. The idea was first developed by Servin et all [SW16] in the nonsmooth case. We adapt their algorithm for the penalty-based DEM. The goal is to derive a parallelized adaptive algorithm of particle dynamics which ideally has only small errors and is real-time capable.

Rigid Body Description As in Chapter 3, we consider a particle system with N particles. The set of particles $\mathcal{N}$ is split into pairwise disjoint subsets $\mathcal{N}_A, \mathcal{B}, \ldots$. A rigid aggregate A consisting of $\mathcal{N}_A$ particles has a total mass

$$m_A = \sum_{a \in \mathcal{N}_A} m_a.$$

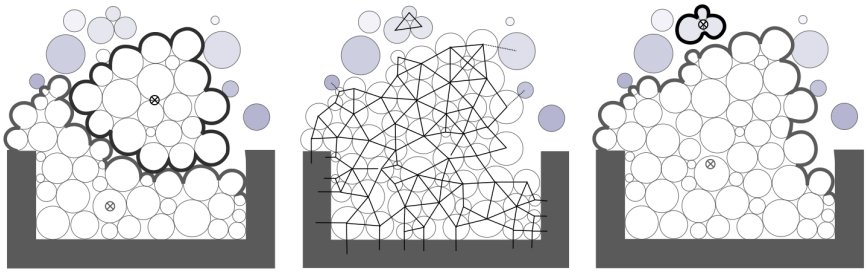


Figure 4.2: Visualization of the Adaptive Particle Merging approach, with kind permission taken from [SW16, Figure 3, p. 112]

The center of gravity $\boldsymbol{x}_A$ is given by $\boldsymbol{x}_A = m_A^{-1} \sum_{a \in \mathcal{N}_A} m_a \boldsymbol{x}_a$ and the aggregate velocity $\boldsymbol{v}_A$ may be written as $\boldsymbol{v}_A = m_A^{-1} \sum_{a \in \mathcal{N}_A} m_a \boldsymbol{v}_a$. Then for each particle we may introduce a relative coordinate system

$$\mathcal{K}_{aA} = \left\{ \boldsymbol{e}_x^a, \boldsymbol{e}_y^a, \boldsymbol{e}_z^a \right\}$$

with the origin relative to the aggregates position $\boldsymbol{r}_{aA} = \boldsymbol{x}_a - \boldsymbol{x}_A$.

Affine Linear Embedding Given a model reduced system, consisting of $n < N$ possibly aggregated particles, we can define the mapping

$$\begin{aligned} \mathbb{P}: \qquad & \mathbb{R}^n \longrightarrow \mathbb{R}^N \\ \boldsymbol{q}_n = (\boldsymbol{x}_A, \boldsymbol{x}_B, \dots) & \longmapsto \boldsymbol{q}_N = (\boldsymbol{x}_a, \boldsymbol{x}_b, \dots), \end{aligned}$$

where $\mathbb{P} = A + \boldsymbol{b}$ with

$$A = \begin{bmatrix} \mathbb{I}_{3\times3} & 0 & \dots \\ \mathbb{I}_{3\times3} & 0 & \dots \\ \vdots & & \\ 0 & \mathbb{I}_{3\times3} & \dots \\ \vdots & \dots & \mathbb{I}_{3\times3} \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{r}_{aA} \\ \boldsymbol{r}_{bA} \\ \vdots \\ \boldsymbol{r}_{cB} \end{bmatrix}$$

Merge Particles Particles being at rest or having similar velocities are merged. That means we form a new rigid body called an aggregate particle of the shape of the combined particles, similar to a multisphere approach. Thereafter, the contact forces between particles of the same aggregate do not need to be evaluated, which reduces computation time.

Split Particles In their paper, they suggest three splitting strategies, see [SW16]. *Contact splitting* considers normal, tangential and rotational velocities. If the forces on particles forming part of a rigid aggregate exceed a certain threshold, the velocities increase and a split is performed. *Trial solve split* is a more academic splitting technique where the full particle system is solved simultaneously. Where the difference to the reduced merged system becomes apparent, a split is performed. From our point of view the most practical splitting technique is the *sensor split*. Around a certain sensor, e.g. the excavation bucket, we always refine and compute the full system without rigid aggregates, while in regions further away of the sensor, merging is performed.

4.2.2 Frozen Particles

This idea is based on the dissertation, see [Obe13]. Instead of always integrating all particles, we just focus on a region of interest, where particles interact with a tool. We divide the set $\mathcal{N}$ of N particles into two sets $\mathcal{N}_A$ and $\mathcal{N}_B$. All other particles in $\mathcal{N}_B$ remain frozen, that is we neither need to compute contact forces nor integrate their dynamics. This may save time, since the bottle neck of a typical DEM simulation is contact detection and time integration. The difficult point is the administration of suitable index sets.

4.2.3 PBD-DEM Coupling

A slight extension of the Frozen Particles approach, is the coupling of DEM with PBD. In each contact time step perform a PBD step for all particles distant from the tool of interest. During finer time steps resolve the particle-tool interaction using classical Discrete Element Contact resolution. The difficulty is the stability in the contact zone, where particles N_B interact with particles N_A, and the efficient handling of the dynamic index sets.

4.3 Data-based Hybrid Models

In this section, we enter the field of data-based algorithms. Here, we tend to apply models that rely upon measured or previously computed data. The advantage is, that the equations in play are much simpler and thus realtime computation is feasible. Major drawback is the lack of physical interpretation and that the models do not yield good results if similar data has not been previously generated and incorporated in the models. From now on, all DEM computations are based on

Algorithm 5 Merge and split algorithm as explained by Servin et al. [SW16]

```
1: initialize q_0, q̇_0
2: for dt < t_end do
3:     if dt_C|dt then
4:         contacts =detectContact(q)
5:         for c in contacts do
6:             if MergeCondition(c) then
7:                 Merge particles
8:             end if
9:         end for
10:    end if
11:    for c in contacts do
12:        force = resolveContact(c)
13:    end for
14:    for p in free particles do
15:        Integrate force
16:        updatePosition(p)
17:    end for
18:    for A in rigid aggregates do
19:        Integrate force
20:        updatePosition(A)
21:        for p in particles of A do
22:            updatePosition(p)
23:        end for
24:    end for
25:    if dt_c|dt then
26:        for A in rigid aggregates do
27:            if SplitCondition then
28:                Split rigid aggregate A
29:            end if
30:        end for
31:    end if
32: end for
```

Algorithm 6 Hybrid PBD-DEM time step

```
1: for all timestep Δt_C do
2:    N_A, N_B = Tool-Distance-search
3:    Search for contact constraints
4:    PBD-step for all particles N_B
5: end for
6: for timestep Δt_DEM do
7:    for all constraints g_k for particles N_A do
8:      Compute DEM-PP-Forces for particles N_A
9:      Compute DEM-PT-Forces
10:   end for
11:   Integrate particles N_A
12: end for
13:
```

the linear penalty-based DEM model presented in the previous chapter in Section 3.4.2. The generation of input data is performed using the software GRAPE with non-rotational particles. The model parametrization from the previous chapter in Section 3.6 is underlaid with specific examples in Section 5.2. For now, we focus on the general description of data-based hybrid models. First, we present the LUT, based upon Section 2.4.1. We present different ideas on TPWL and stochastic processes. Second, we present RNNs as a means to assess excavation forces in realtime. Both models rely on previous offline simulation of basis maneuvers. Thereafter, we build the online model, which can be used in realtime, see also Figure 4.3.

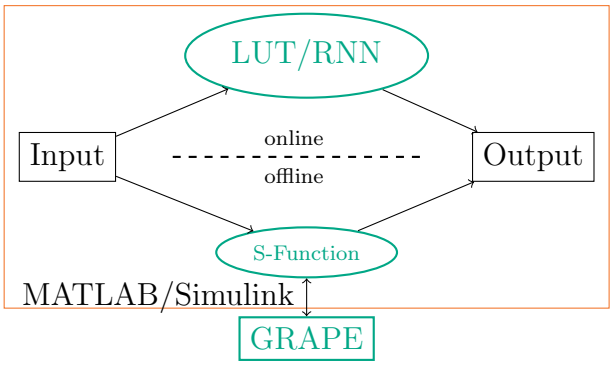


Figure 4.3: Program structure of the hybrid data-based approach: in the lower half, the classical DEM framework used in the offline phase; in the upper half, the online phase using the DEM LUT or the DEM RNN approach .

4.3.1 DEM LUT Approach

The main idea is to define a set of parameters $\mathcal{P}$, that describes the state of a soil-tool interaction. In a first offline phase, full DEM simulations are performed to obtain data for a discrete set of parameters $\boldsymbol{p}_i \in \mathcal{P}$. The data is collected in a LUT, see Subsection 2.4.1. In a second online phase, we access the collected data by finding the closest neighbor $\tilde{\boldsymbol{p}}$ to the current parameter setup $\boldsymbol{p}$. Similarly, we can calculate some kind of weighted mean or even generate a stochastic process. Parts of the following results have been published in the conference proceedings [JSB19; Jah+20] and are for the sake of completeness summarized and reproduced in the following.

Soil-tool interaction forces arising at an earthmoving tool are highly nonlinear. In fact, when we consider measurements of soil-tool interaction of the same maneuver with similar input, the output may change drastically due to different drivers and slightly differing trajectories [Bal+16, Fig. 11, p. 9]. Whence, it is reasonable to approximate this highly nonlinear behavior using LUTs, at best with similar frequency and magnitude characteristics.

To achieve real-time capable simulation speed, we use a two-phase computation method. In an offline phase, time-consuming DEM simulations are executed, thereby varying a set of tool-parameters. Thereupon, forces and moments are extracted and saved in a LUT. The subsequent online phase consists in finding a meaningful approximation for a given soil-tool state, using the LUT data. More specifically, we perform a set of DEM simulations varying some relevant tool parameters, e.g. the tool position, velocity and rotation. The particle properties representing the parametrization of a specific soil remain unchanged. The DEM simulations are executed in parallel on a high-performance computing cluster. The parameter variation is performed in a loop such that the table data is systematically generated.

The current soil model, is based upon the DEM LUT approach incorporating many Discrete Element simulations performed in an offline phase. Each LUT entry corresponds to a certain digging maneuver which is characterized by the material, and the geometry of the particle pile. A single simulation in one LUT corresponds to a certain tool configuration, that is a tool parameter set. Currently, we consider the speed in longitudinal direction, the angle of incidence and most importantly the cutting depth. Each LUT entry corresponds to the postprocessed output of a full DEM simulation, extracting the force and moment time series. We usually forget the initial and final phase, as artifacts may arise here. This is due to the fact, that initially the tool may not be in contact with the particles. Towards the end of the simulation, the digging tool crushes particles into the boundary resulting in an overestimation of digging forces and torques. In between, we compute the

Figure 4.4: Visualization of numerical example of plate in gravel trench

expectation value and the variance of the time series. This information suffices to generate a signal with similar overall behavior and the amplitude of the oscillations may be preserved.

Within a single basis simulation, i.e. when filling one LUT entry, the respective tool parameters remain constant. The results, i.e. the force and moment time series are gathered and reduced to expectation value and standard deviation per component, containing the main information for a specific tool state. We cut off the initial part of the time series, where the tool enters into the soil and also the final part of the time series where the surcharge, i.e. the accumulated soil in front of the tool is increasing. Due to the fact that we require at least ten particles between tool and boundary to obtain a meaningful force output, we make sure, that within the final simulation state, the tool is fully surrounded by particles and far enough from reaching the boundary. The resulting data structure resembles a multi-dimensional table, where one entry, consisting of mean forces, moments, belongs to a specific set of tool parameters. We generate a second table with the respective standard deviations of the time series, in order to capture the oscillations.

In the online phase, the tool is moved and the relevant parameters, namely position, longitudinal velocity and rotation are gathered to find a good approximation using the data from the LUT. We achieve this, performing a k-nearest neighbor search and using weighted means [Bar12, Chapter 14, p. 317] and also Subsection 2.4.1.

Numerical Example: Plate in Trench As a first example, we present a rectangular plate moving through gravel, which has been presented in previous work [Obe+11; KOB13], see Figure 4.4. Experiments have been performed in cooperation with the soil laboratory at Technical University Kaiserslautern, thus the accuracy of our simulation results can be assured by comparison to the measurement data, see Section 5.3. In this example, the parameter space is of dimension three. More

specifically, we modify the cutting depth d in vertical z-direction. Furthermore, the angle θ around the global lateral y-axis, as well as the plate's speed v in longitudinal x-direction are varied, compare also Figure 4.5. This choice is motivated by earthmoving equations, where the velocity and the cutting depth d have a quadratic influence on the force, see Section 4.1 and the references [Obe+11; McK85; Ree64; WD07]. The findings in 5.3 revealed, that the influence of the cutting depth indeed might be quadratic, but that the quadratic velocity term presented in the FEE overestimates the forces for our specific application. We experienced, that the influence of the angle of incidence is nonlinear. Let us recall the force equation with the terms of interest from Equation (4.5), which reads

$$F = (\gamma g d^2 N_\gamma + \gamma g v^2 d N_v + q d N_q) w,$$

depending on the the bulk density γ, the gravity g, cohesion c, surcharge q and cutting width w. The coefficients N_γ for the passive earth pressure, N_v for the velocity and N_q for the surcharge depend on different angles via trigonometric identities, see Section 4.1. This motivates the parameter choice of the LUT. Both the cutting depth d and the velocity v have a quadratic influence. The angle of incidence $\theta = 90 - \alpha$ influences all N-factors. Hence, the LUT parameters are chosen accordingly. The initial study of the FEE, leads to the choice of three Lookup parameters, namely the cutting depth d, the x-velocity v_x, and the angle around the y-axis θ. An assessment of whether to incorporate the velocity v_z in z-direction was made at a later point. The LUT can be written as a nonlinear mapping

$$\begin{aligned} L_{LUT}: \quad & \mathcal{P} \longrightarrow \mathcal{F}, \\ & (d, \theta, v) \longmapsto (\boldsymbol{F}, \boldsymbol{T}) \end{aligned} \tag{4.6}$$

with parameter space $\mathcal{P} \subset \mathbb{R} \times [0, 2\pi] \times \mathbb{R}^+$ and $\mathcal{F} \subset \mathbb{R}^6$. $\boldsymbol{F}$ denotes the force vector, while $\boldsymbol{T}$ stands for the torque vector. Furthermore, we generate a second table regarding the oscillations of the basis simulations in terms of the standard deviation of the time series, namely

$$\begin{aligned} L_{LUT_\sigma}: \quad & \mathcal{P} \longrightarrow \mathcal{F}_\sigma, \\ & (d, \theta, v) \longmapsto (\boldsymbol{F}_\sigma, \boldsymbol{T}_\sigma), \end{aligned}$$

where $\boldsymbol{F}_\sigma$ corresponds to the standard deviation of the measured force vector time signal. Analogously, $\boldsymbol{T}_\sigma$ stands for the standard deviation of the measured torque vector time signal.

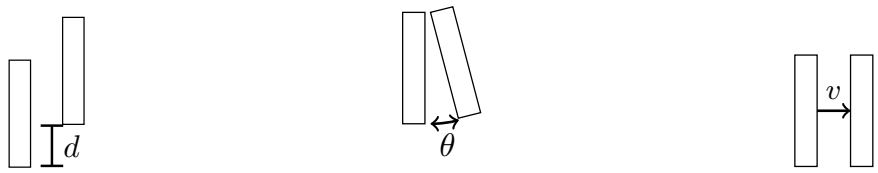


Figure 4.5: Visualization of the variation of different tool parameters cutting depth d, angle of inclination θ and longitudinal velocity v

Parameter	Cutting depth	Angle	Velocity
Symbol	d	θ	v
Unit	m	$^\circ$	m/s
Interval	$[-0.25, -0.05]$	$[-45, 75]$	$[0.5, 2.0]$
Discretization step	0.1	30	0.5

Table 4.1: Computing requirements for offline phase

Offline Phase During a time-consuming offline phase, we perform full DEM simulations for different tool-parameter configurations within a MATLAB/Simulink environment, see [Mat] and also Figure 4.3. The parameter range for our example LUT can be observed in Table 4.1. The simulation time for one table entry corresponds to two seconds, in total we perform 60 DEM simulations. The number of basis simulations is problem specific and depends on the chosen parameters and on the desired accuracy. The force sampling rate corresponds to 1 kilohertz. The acquired time series are then processed and the expectation value and the standard deviation are computed. Therefore, we cut off an initial starting phase and a final phase of about one second simulation time, in order to avoid boundary artifacts. In Figure 4.6, the sensitivity of the mean force in longitudinal x-direction with respect to rake angle θ and cutting depth d can be observed.

Online Phase In the online phase, see Figure 4.3, we load the LUT into the MATLAB workspace. We describe four different variants for the online phase. In each time step, within our example the tool parameter vector $\boldsymbol{p} = (d, \theta, v)$, is used to search for a close approximation from the LUT. The parameter vector $\boldsymbol{p}$ is obtained from the current tool position $\boldsymbol{x}$, the tool velocity $\boldsymbol{v}$, the current orientation represented by a quaternion $\boldsymbol{q}$. Let us write $\mathcal{F} = [\boldsymbol{F}, \boldsymbol{T}]$ for the generalized force covering both force and torque vector.

Variant 0 (Lookup 0): As described in Section 2.4.1, we compute

$$\tilde{\boldsymbol{p}} = \underset{\boldsymbol{p}_i \in \mathcal{P}}{\arg\min} \|\boldsymbol{p} - \boldsymbol{p}_i\|_2.$$

Then we compute the expected force and torques by using the LUT $\mathcal{F} = L_{LUT}\left(\tilde{\boldsymbol{p}}\right)$.

Variant 1 (Lookup 1): Again, we compute the nearest neighbor

$$\tilde{\boldsymbol{p}} = \underset{\boldsymbol{p}_i \in \mathcal{P}}{\arg\min} \|\boldsymbol{p} - \boldsymbol{p}_i\|_2.$$

The Jacobian of our LUT L_{LUT}, compare Equation (4.6) by

$$J_{L_{LUT}} := \frac{\mathrm{d}L_{LUT}}{\mathrm{d}\boldsymbol{p}} = \begin{pmatrix} \frac{\partial F_x}{\partial d} & \frac{\partial F_x}{\partial \theta} & \frac{\partial F_x}{\partial v} \\ \frac{\partial F_y}{\partial d} & \frac{\partial F_y}{\partial \theta} & \frac{\partial F_y}{\partial v} \\ \frac{\partial F_z}{\partial d} & \frac{\partial F_z}{\partial \theta} & \frac{\partial F_z}{\partial v} \\ \frac{\partial T_x}{\partial d} & \frac{\partial T_x}{\partial \theta} & \frac{\partial T_x}{\partial v} \\ \frac{\partial T_y}{\partial d} & \frac{\partial T_y}{\partial \theta} & \frac{\partial T_y}{\partial v} \\ \frac{\partial T_z}{\partial d} & \frac{\partial T_z}{\partial \theta} & \frac{\partial T_z}{\partial v} \end{pmatrix}$$

can be approximated as follows. Let the index of the nearest neighbor

$$s_1 = \underset{i \in \{1,\ldots,N\}}{\arg\min} \|\boldsymbol{p} - \boldsymbol{p}_i\|_2$$

and second nearest neighbor

$$s_2 = \underset{i \in \{1,\ldots,N\} \setminus \{s_1\}}{\arg\min} \|\boldsymbol{p} - \boldsymbol{p}_i\|_2.$$

Then a finite difference approximation of $J_{L_{LUT}}$ can be computed and we may write

$$\mathcal{F}\left(\boldsymbol{p}\right) \approx L_{LUT}\left(\tilde{\boldsymbol{p}}\right) + J_{L_{LUT}|\tilde{\boldsymbol{p}}}(\boldsymbol{p} - \tilde{\boldsymbol{p}}).$$

Variant 2 (SDE): As in the previous variants, we compute the nearest neighbor

$$\tilde{\boldsymbol{p}} = \underset{\boldsymbol{p}_i \in \mathcal{P}}{\arg\min} \|\boldsymbol{p} - \boldsymbol{p}_i\|_2.$$

We compute the force expectation value $\mu_F := L_{LUT}\left(\tilde{\boldsymbol{p}}\right)$ and the standard deviation $\sigma_F := L_{LUT_\sigma}(\tilde{\boldsymbol{p}})$ using the LUT componentwise. For the torques, the procedure is analogous. Then we compute for each time step the normally distributed random

variable $F_t \sim \mathcal{N}(\mu_F, \sigma_F)$. This can be interpreted as a stochastic process described by a Stochastic Differential Equation (SDE) of the form

$$\mathrm{d}F_t = (\mu_F(t) - F_t)\mathrm{d}t + \sigma_F(t)\mathrm{d}W_t, \quad F_0 = 0, \tag{4.7}$$

see Section 2.3 for the definition of Brownian Motion W_t. The first term on the right hand side of equation (4.7) covers the transition probability from the previous mean $\mu_F' := (L_{LUT}(\tilde{\boldsymbol{p}}')$ of a Lookup query to a current mean $\mu_F := L_{LUT}(\tilde{\boldsymbol{p}})$. The second term generates noise resulting in an oscillatory behavior, see Figure 4.7. The stochastic differential equation (4.7) bears similarity to an Ornstein-Uhlenbeck process given by

$$\mathrm{d}X_t = \eta(\mu - X_t)\mathrm{d}t + \sigma\mathrm{d}W_t, \quad X_0 = a.$$

Solution theory on SDEs can be found in [KS05, Section 5.2, p. 284, Section 5.6, p. 354], but a further analysis is beyond the scope of this thesis. The idea came to me as the Lookup 0 approach, on the one hand, yields piece-wise constant output. The DEM, on the other hand leads to oscillations in the tool forces which depend on several aspects. First, the tool mesh has an effect, very sharp triangles lead to artifacts in the collisions. Second, the damping parameters, but also the time steps have an influence on the oscillations. Third, the order of the particle indexation has an influence when parallelized codes are used. Fourth, the nature of considering the bulk behavior of particles ignore the fact, that each particle trajectory may slightly differ and have a chaotic impact on the bulk behavior. More specifically, a slight change in the initial conditions may, after several time steps, lead to a collision between particles which otherwise may not have occurred. All these reasons, give rise to a model inherent property of possibly stochastic oscillations in the tool forces. On the other hand, when coupling the soil force prediction model with a multibody system, oscillations lead to unstable behavior. This is why, we stopped looking into further details of this variant incorporating SDEs as this is not relevant for our application.

Variant 3 (TPWL): The TPWL approach has been introduced in Section 2.4. Instead of computing only the nearest neighbor, we might compute several neighbors and interpolate using weights, which depend on the distance of the respective neighbor. For $\boldsymbol{p} \in \mathbb{R}^3$, we may compute up to eight neighbors until we have selected all surrounding neighbors. Or in general for $\boldsymbol{p} \in \mathbb{R}^n$, we may compute up to 2^n directly surrounding neighbors. Thus we obtain $\boldsymbol{p}_1, \ldots \boldsymbol{p}_k$ with $k \leq 2^n$ Then we compute the generalized force by

$$\mathcal{F}(\boldsymbol{p}) \approx \sum_{j=1}^{k} w_j L_{LUT}(\tilde{\boldsymbol{p}}_j),$$

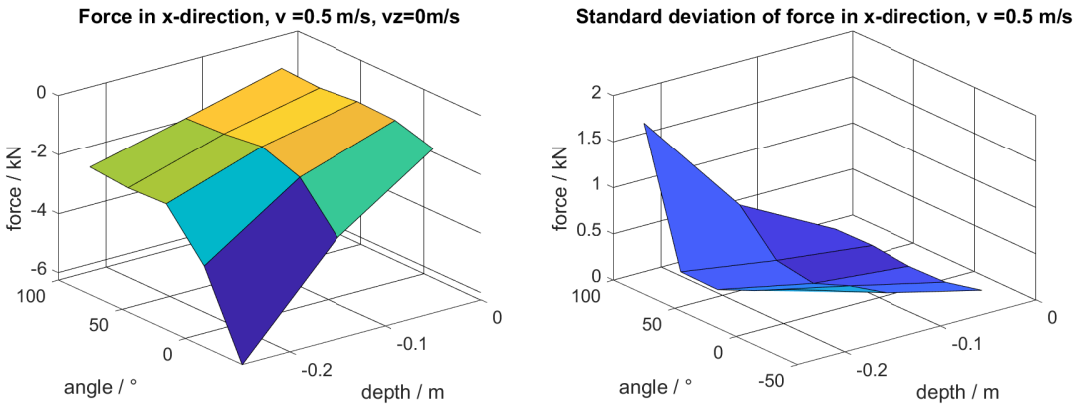


Figure 4.6: Expectation value and standard deviation of force magnitude in x-direction for different depth d and angle θ for fixed velocity $v = 0.5$ m/s

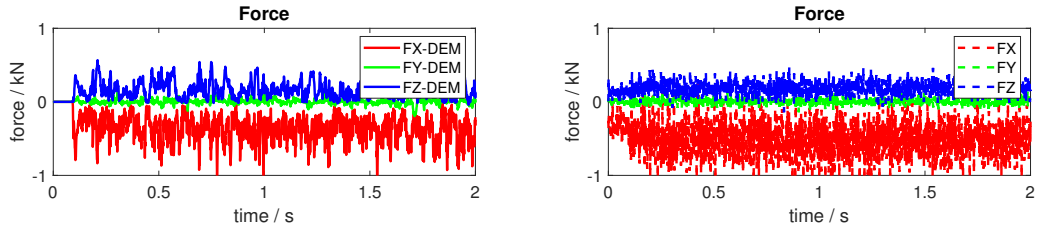


Figure 4.7: Force time series for parameters $d = -0.05$ m, $\theta = -15$ degrees, and $v = 1.5$ m/s with GRAPE (left) and random DEM LUT approach (right)

where w_j describes the distance dependent standardized weights, i.e. $\sum_{i=1}^{k} w_j = 1$. TPWL can also be combined with the Lookup 1 or the SDE variant.

In summary, the online phase of the DEM LUT approach consists in performing a nearest neighbor search. Consequently, we compute force and torque vectors, according to one of the presented variants, see Figure 4.7 for a trial simulation based on stochastic processes. Our numerical experiments show that loading the Lookup data-structure takes up most of the time. Once, the offline data is collected and the simulation scenario is set up, we are capable of accessing the data in realtime. In the remainder of this thesis, we will often abbreviate DEM LUT approach by LUT, especially in legends of plots where brevity is essential.

4.3.2 DEM RNN Approach

The DEM LUT approach yields convincing results for academic test cycles of dragging a plate or a bucket through a soil trench. However, when looking at MBS models of a full excavator, the excavation cycles become much more complex. The approximation by our DEM LUT approach often does not cover all aspects of a maneuver. We had to make the choice of either enlarging the parameter space, by varying also the vertical velocity v_z, the angular velocity ω_y, etc. or implementing a different model. In the former approach, due to the curse of dimensionality, we would have to add hundreds of offline basis simulations, if further parameters were added. So we went for the latter and studied the approximation behavior of RNNs. At first, one might think that a simple FNN suffices, but RNNss have several advantages. Due to the hidden states and the recurrent structure, previous time steps influence the current output. This bares similarities to a dynamical system. RNNs are known to yield good results in time-dependent processes, e.g. in text recognition or with time series and biological data, see [Agg18, Chapter 7, p. 271]. As FNN, they possess the Universal Approximation property, see Subsection 2.4.3, and are known to be Turing complete.

Input Selection The application of RNNs in the context of DEM is described in the following, similar to the DEM LUT approach we coin it DEM RNN approach. The main difficulty poses the selection of suitable training data and the definition of the net architecture. At hand was the simulated data used for the LUT. As input, we chose the full generalized position vector and its derivative, comprised of $\boldsymbol{q} = (\boldsymbol{x}, \boldsymbol{p})$ consisting of position vector $\boldsymbol{x}$ and quaternion $\boldsymbol{p}$ and its derivative $\dot{\tilde{\boldsymbol{q}}} = (\boldsymbol{v}, \boldsymbol{\omega})$, compare Table 2.1. This results in a total of 13 inputs. However, most of these inputs are zero, as we change only the cutting depth, the longitudinal velocity and the angle of incidence in the Lookup maneuvers. The trajectories are thus situated in the $x - z$ plane. Consequently, two translational degrees of freedom and one rotational degree of freedom suffice for the motion description. Consequently, we consider $\boldsymbol{x}_x$, and $\boldsymbol{x}_z$, two non-zero entries in the quaternion $\boldsymbol{p}$, the velocities $\boldsymbol{v}_x$ and $\boldsymbol{v}_z$ and the angular velocity $\boldsymbol{\omega}_y$, that is a total number of 7 inputs.

When the total number of inputs is chosen as 13, we often run into problems using the scaling variant 1, see Paragraph 2.4.3.

Instead of using the Lookup data, full digging trajectories of different excavator models were used. First, we tried to generate realistic maneuvers with a simplified model of an excavator. However, these trajectories did not contain oscillations and were still far from the trajectories expected at the driving simulator. That is

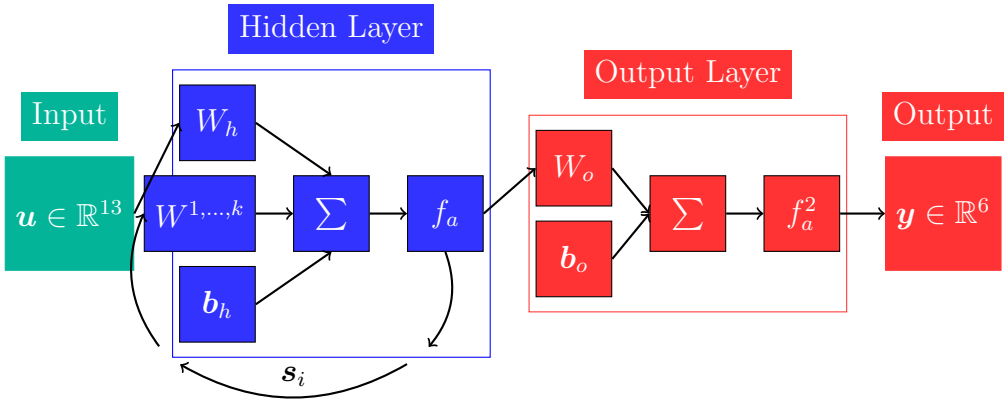


Figure 4.8: Visualization of a RNN with 13 input nodes, ten hidden nodes ($W_h \in \mathbb{R}^{10\times13}$), one hidden layer, $k = 5$ recurrent layers and six output nodes

why we recorded excavation trajectories and used these for offline simulations, see Section 5.4.

Output The most relevant forces are the longitudinal force F_x and the vertical force F_z in global coordinates. The torque T_y is the third relevant output value. The force term F_y and the two additional torques, are expected to have only small impact. However, we aim at the construction of a prediction algorithm for all six force torque outputs.

Structure of the RNNs With the previous thoughts in mind, different network structures are reasonable. First, one could train six separate RNNs with one dimensional output variables y. Or one network with six dimensional output $\boldsymbol{y}$. Second, the number of neurons per hidden layer N_n can be adapted. Third, the number of recurrent layers N_r, containing previous states, i.e. information on previous time steps has to be chosen. If we set N_r to zero, we obtain a FNN. The number of hidden layers N_h rapidly increases the degrees of freedom. During training, backpropagation will first change the weights in the layers close to the output. If $N_h > 1$, this leads to longer training times. The proofs on Universal Approximation also indicate that one large hidden layer often suffices for small applications.

MATLABs Layrecnet Implementation After the training is completed, the relation between input $\boldsymbol{u}$ and output $\boldsymbol{y}$ is known as a nonlinear function of the form

Algorithm 7 RNN loop, compare to Figure 4.8

```
for all timesteps do
    s_t = f_a(Σ_{i=1}^{ks} W^i s_{t-i} + W_h u_t + b_h)
    for i = 1 : ks do
        s_i = s_{i-1}
    end for
    y_t = f_a^2(W_o s_t + b_o)
end for
```

Discussion In total, this leads to good approximations of forces and moments, while the visualization is lost in the post-processing step. The idea is to couple Unity with the multibody excavator model and the LUT or the RNN for real-time force prediction. For an initial test-setup, we implement the coupling of the excavator with GRAPE.

5 Industrial Applications and Experiments

In the previous chapter, we presented different approaches for realtime force prediction. The most promising approaches discussed, are the data-based models, that is a LUT and RNN of Section 4.3. The aim of this chapter is to apply these methods on an excavator simulator. But first, we will describe the general procedure for online soil tool interaction in Section 5.1. Thereafter, we will look at the proper parametrization of particle samples in Section 5.2. In addition, we will present a validation of the DEM code, based upon measurements performed at a test pit at TUK soil laboratory in Section 5.3, which is closely related to the study of the LUT from Section 4.3.1. Finally, we describe the development process of the methods at the RODOS and present several numerical examples, see Section 5.4.

5.1 Soil-Tool Interaction in Realtime

In the view of industrial application, we want to describe our method in a more general context. That means, that we want to illustrate the procedure of setting up realtime soil-tool interaction for a general earthmoving application.

Given a soil sample and an agricultural or construction application concerning soil-machinery interaction, we first need to be able to carry out a DEM simulation of the given application. Therefore, we perform a model parametrization of a reference soil sample, as described in Section 3.6, specific examples are presented in Section 5.2.

We require a realistic multibody model for the trajectory generation of the earthmoving machinery. On the one hand, we need the model to specify the working range of the earthmoving tool. On the other hand, this allows for the computation of realistic trajectories and operator dependent variations within them. If we can classify the trajectories by few meaningful parameters, the LUT is applicable. If the parameter space is large, because of the curse of dimensionality, the generation

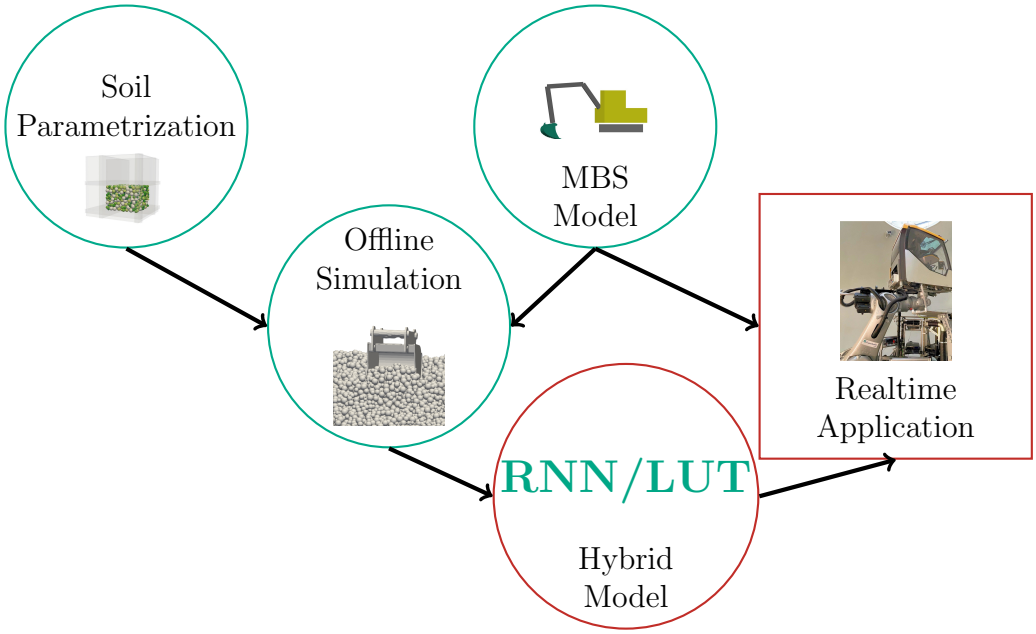


Figure 5.1: Sketch on the general procedure for realtime soil-tool interaction as described in this thesis

of enough basis simulations becomes too expensive. In that case, RNNs are less restrictive, since fewer relevant training trajectories suffice for the model setup.

Either way, we need to perform offline simulations to collect data for the generation of a realtime capable hybrid model. With the MBS model of the construction machinery, the generation of comprehensive trajectories discretizing the working range of the tool is simplified. The generated tool trajectories form the foundation for basis simulations or training data and can be computed in parallel on a CPU-Cluster. Depending on the total simulation, the particle size, the available computation resources, and the number of particles, the generation of one DEM basis simulation typically lasts between one hour up to several days.

When the offline data is precomputed, we generate a hybrid model for realtime force prediction as described in the previous chapter in Section 4.3. If we can classify the variation of the machine maneuvers while interacting with soil by several specific parameters, we can apply the DEM LUT approach. Otherwise, it might be more suitable to work with the more general DEM RNN approach. Here, we require a high number of realistic maneuvers with all probable trajectory variations. The procedure is visualized in Figure 5.1.

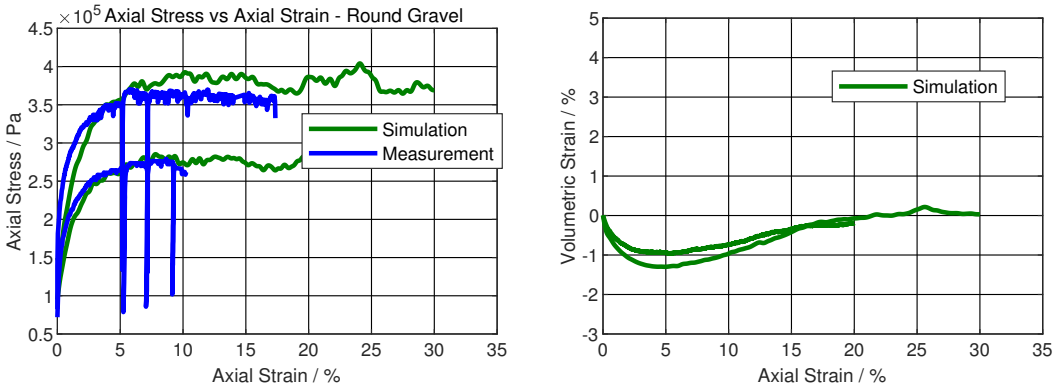
Symbol	Name	Range	P^{rg}_{TT}	P^{cs}_{TT}	P^{sg}_{TT}	Unit
Sample parameters						
r_i	radius	$0.1 - 100$	$8 - 16$	$10 - 20$	$25 - 60$	mm
n	porosity	$0.3 - 0.6$	0.37	0.33	0.44	–
ρ	density	$1000 - 3500$	2700	2680	2650	$\mathrm{kg/m^3}$
Normal interaction						
E	Young modulus	$10^5 - 10^8$	$3e7$	$1.2e8$	$4e7$	$\mathrm{N/m^2}$
D_N	damping coefficient	$[0 - 1]$	0.1	0.1	0.1	–
k^N_{ij}	normal stiffness	$\frac{\pi}{2}\frac{r_i+r_j}{2}E$				N/m
d^N_{ij}	normal damping	$D_N 2\sqrt{\tilde{m}_{ij}k^N_{ij}}$				Ns/m
Tangential interaction						
E_T	tangential modulus		$7e6$	$1e8$	$4.8e7$	N/m
k^T_{ij}	tangential stiffness	$\frac{\pi}{2}\frac{r_i+r_j}{2}E_T$				N/m
D_T	damping coefficient	$[0 - 1]$	0.1	0.08	0.08	–
d^T_{ij}	tangential damping	$D_T 2\sqrt{\tilde{m}_{ij}k^N_{ij}}$				Ns/m
Coulomb friction						
μ	friction coefficient	$[0 - 1]$	0.28	0.23	0.25	–

Table 5.1: Parameters of the three cohesionless materials within GRAPE

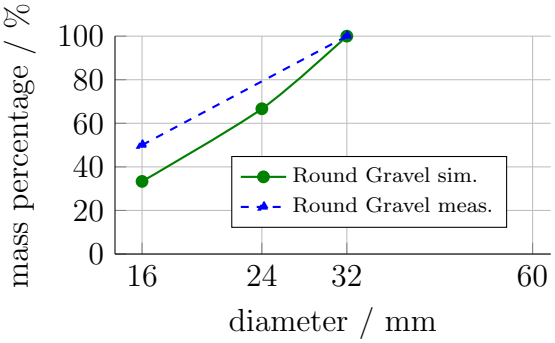
5.2 Parametrization of Specific Soil Samples

In this section, we want to describe the specific parametrization procedure for GRAPE. This parametrization procedure was developed in the PhD-thesis [Obe13] and the references therein. In the context of this thesis, there are three relevant cohesionless materials, for which we briefly present the parametrization results. We performed TTs with different sidewall pressures, as discussed in Section 3.6. We apply a fifth order Butterworth filter with cutoff frequency of 20 Hz. The simulated strain stress curves are all filtered in the graphs presented on the left of Figures 5.2, 5.3 and 5.4.

Parametrizations of Round Gravel This material was used in the first experiments presented in the following Section 5.3. Round gravel with a diameter between 1.6 and 3.2 cm, was measured in a small triaxial cell. The sidewall pressure was in the range of 64 and 87 kPa. The grain size distribution was not measured specifically, but ranges within the minimum and maximum diameter. The particle size distribution was chosen to correspond to the grains, that is we did not scale the



(a) Strain-stress characteristic: Measurement and simulation results for a *round gravel* specimen (left) with particle interaction parameters P_{TT}^{rg} for sidewall pressures of 64 kPa (left-bottom curve) and 87 kPa (left-top curve). The measurement of the volumetric strain was not succesful (right), we just present the results from the simulations.

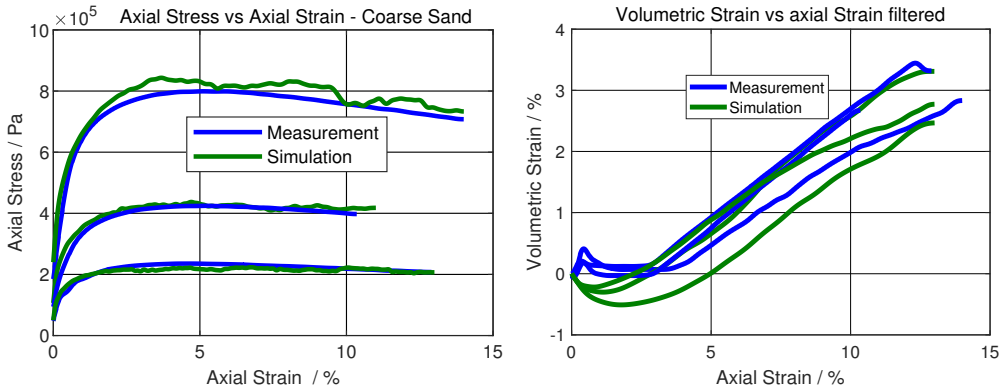


(b) Grain size distribution: *polydisperse* distribution for *round gravel* chosen in the simulation (**green**). The grain size diameter lies between 16 and 32 millimeters.

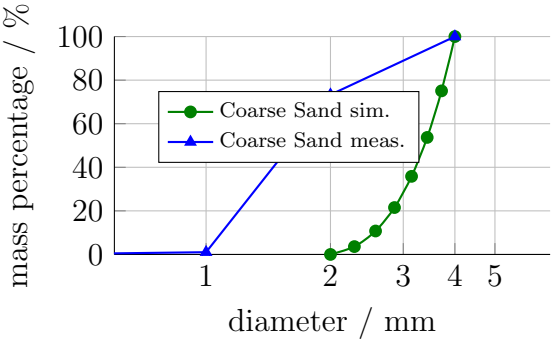
Figure 5.2: Parametrization of Round Gravel Sample

particle size. The dents in the measured curve on the left of Figure 5.2 arise due to unloading phases in the large triaxial cell. No data on the measured volumetric strain characteristic is available. On the right of Figure 5.2, we just present the simulated volumetric strain. We present the parameters P_{TT}^{rg} in Table 5.1.

Parametrizations of Coarse Sand The second material for the testbed was coarse sand with grain diameter of 1 to 4 mm. The sidewall pressure was in the range of 54 and 204 kPa. The material was already presented in [Ste+21]. Here, we scale up the particle size by a factor of two, see Figure 5.3 on the right. The volumetric strain characteristic was similar for all three pressure levels. We present



(a) Strain-stress characteristic: Measurement and simulation results for a *coarse sand* specimen (left) with particle interaction parameters P^{cs}_{TT} for sidewall pressures of 53 kPa (left-bottom curve), 103 kPa (left-middle curve), 203 kPa (left-top curve). The measurement and simulation of the volumetric strain (right)

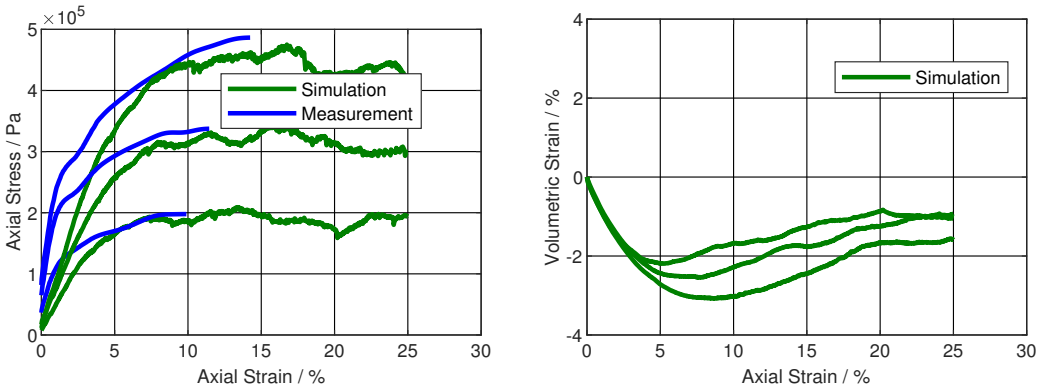


(b) Grain size distribution: *polydisperse* distribution for *coarse sand* – measured in the soil laboratory (**blue**), chosen in the simulation (**green**)

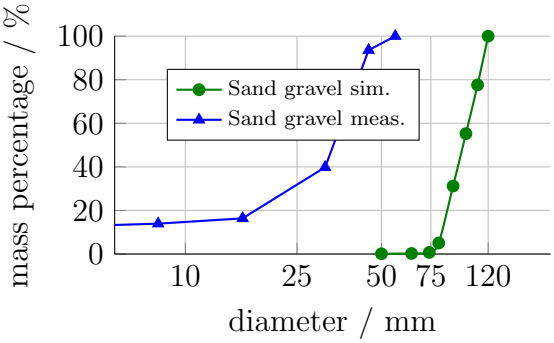
Figure 5.3: Parametrization of Coarse Sand Sample

the parameters P^{cs}_{TT} in Table 5.1.

Parametrizations of Sand-Gravel Mixture We use broken gravel parametrization in the study of excavation in real time, that is mainly in Section 5.4. The results in Figure 5.4 illustrate that we upscaled the particle diameter. The grain diameter reached from fine material to gravel of maximum 5.6 cm. The rather coarse radii in the simulation allow larger time steps, which increases the overall performance in the generation of offline simulations. We present the parameters P^{sg}_{TT} in Table 5.1.



(a) Strain-stress characteristic: Measurement and simulation results for a *Sand-gravel mixture* specimen with particle interaction parameters P^{sg}_{TT} for sidewall pressures of 35 kPa (left-bottom curve), 60 kPa (left-middle curve), and 85 kPa (left-top curve). The measurement of the volumetric strain was not successful (right), we just present the results from the simulations.



(b) Grain size distribution: *polydisperse* distribution for *sand gravel mixture* – measured in the soil laboratory (**blue**), chosen in the simulation (**green**)

Figure 5.4: Parametrization of Sand Gravel mixture Sample

5.3 Bucket Trench Experiment

We use the bucket trench experiment as a benchmark to deepen the understanding regarding relevant soil tool parameters. We also assessed the usage of experimental data as training data for our models. The experimental studies performed in [Obe+11; Obe13] and simulated in the context of [Obe+11; Obe13; KOB13; Kle15b] serve as a point of departure, see also Figure 4.4. First, we analyzed the data from [Obe13] using a plate dragged at different cutting depth, velocities, and inclination. We performed offline simulations for a first LUT design that we published in [Jah+19]. Thereafter, we procured a small excavation bucket, with a cutting depth of 30 centimeters, three mountable teeth and a total weight of 31.6 kg and set up a new measurement campaign at the trench site. In a first attempt, we used round gravel, which resulted in high oscillations and pushed the experimental machinery to its limits. In a second, more systematic and slightly more sophisticated measurement study, we used coarse grained sand. The most relevant results are presented in this section and we compare them to our DEM simulations as a further validation step for GRAPE. The goals of the measurement was the study of the FEE Equation and its parameters, the experimental investigation of the Lookup parameters, and the further validation of the DEM software GRAPE.

Experimental setup We perform the measurements in a soil laboratory with a trench filled with soil. The test site includes a power controlled motor to move the earth moving tool in longitudinal direction. We mount the acquired bucket as an earthmoving tool. Therefore, we used four metal sheets, two at each side of the excavation bucket, in order to fix the bucket below the force measurement box, see Figure 5.6. The force measurement box is equipped with five high-precision force sensors, two in horizontal direction and three in vertical direction. The power controlled motor moves the equipment with a linear guide on an H-beam with

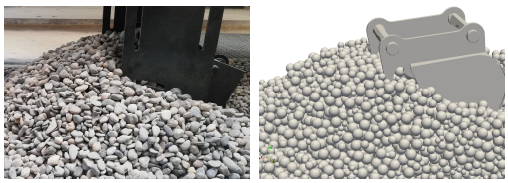


Figure 5.5: Visualization of measurement and simulation for $d = -0.15$ m after dragging the excavator bucket about 2 m through a gravel testbed. From left to right: measurement and full DEM simulation

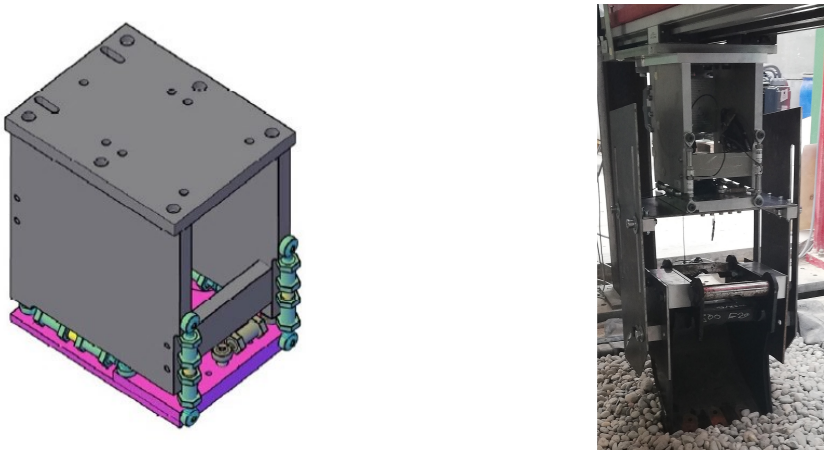


Figure 5.6: Setup of force measurement of small excavation bucket in trench experiment

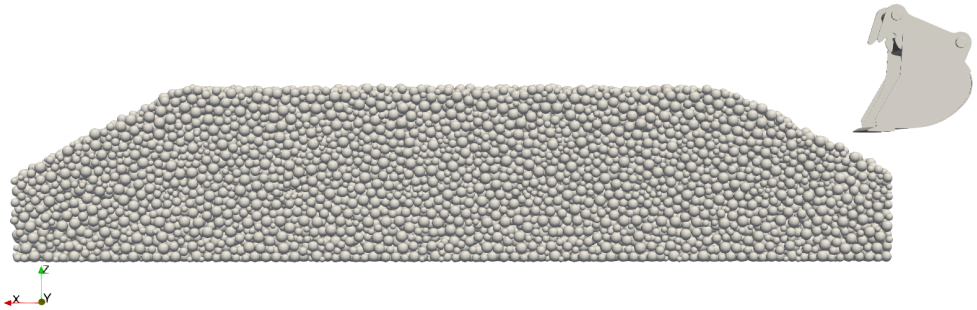


Figure 5.7: Setup for the bucket trench experiment

constant velocity. The metal sheets allow to continuously adapt the cutting depth of the bucket, and also the angle of incidence in discrete steps.

Comment on error estimation of measurements When repeatedly measuring quantities, there are two error sources. The statistical error ε_{stat} arises because each measuring device only has a finite accuracy and slight fluctuations or randomness in the measured quantity are to be expected. The statistical error term can be reduced by means of statistical analysis and by repeating the measurement a number of times. When measuring a time series, these random fluctuations are prominent and lead to oscillations. These oscillations can be analyzed in modal space by Fourier transformation looking at the Power Spectral Density (PSD), see [SM05]. Or, if the time series has a constant level, the computation of mean value μ and standard deviation σ is justified. The standard deviation σ is a second statistical error term,

Parameter	Cutting depth	Angle	Velocity
Symbol	d	θ	v
Unit	m	$^\circ$	m/s
Values	$[-0.15, -0.1, -0.05]$	$[-25, 0]$	$[0.04, 0.12, 0.2]$

Table 5.2: Measurement parameters for coarse sand trench experiment

which is incorporated into the statistical error term. The systematical error ε_{sys} is more difficult to estimate, and can occur repeatedly in each measurement. If a systematical error term is known, we correct it by adding an offset to the measured quantity. However, there always remains an error term. To summarize, if we have a quantity $\boldsymbol{y}$ which is measured, we obtain

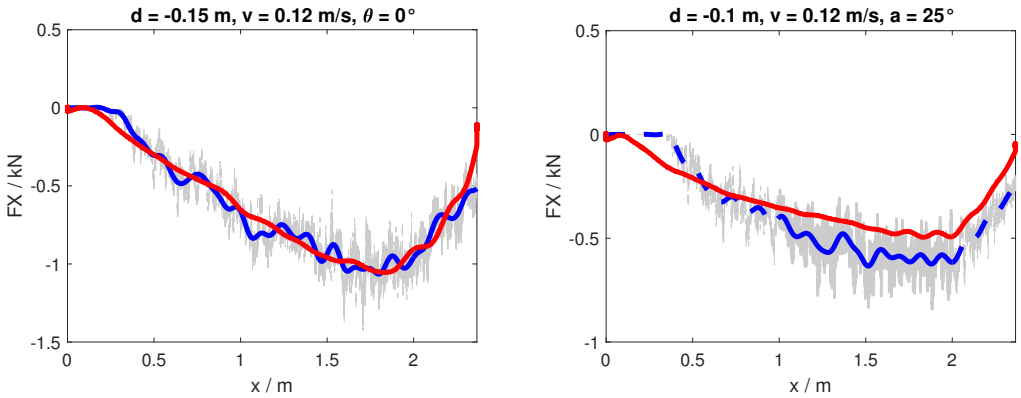
$$\boldsymbol{y} \pm \varepsilon_{stat}(\boldsymbol{y}) \pm \varepsilon_{sys}(\boldsymbol{y}).$$

Measurements We used the test rig to evaluate the influence of the cutting depth d, the velocity v in longitudinal direction and also the tool angle θ. For this study, we filled the trench with coarse sand. In order to capture also the soil cutting behavior of the bucket, we inclined the beginning and the end of the trench by the angle of repose in the experiment and the simulation, see Figure 5.7. Each measurement was performed three times so we estimate the standard deviation of the statistical error term σ_{stat}.

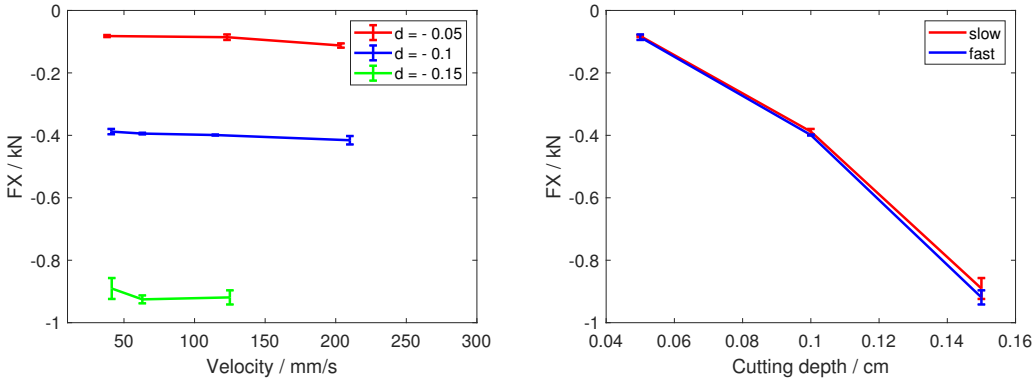
Simulation For the simulation, we used the parametrized sample, see Section 5.2. The simulated test rig has the same dimensions as in the experiment, see Figure 5.7.

Summary of the Experiments The conducted experiments demonstrate, that the cutting depth has clearly the biggest influence on the draft forces of an excavator bucket. The velocity is much less relevant than expected by the FEE, see Section 4.1. Two measurements were performed regarding different angles of incidence. The DEM simulations overall showed good agreement with the measurements, see Figure 5.8b. For small cutting depths, we overestimate the absolute value of the force in longitudinal x-direction. This could be due to a systematic error in the computation of the initial cutting depth.

Performing the experiment was clearly more cumbersome than doing a DEM analysis. Also, while in the measurement campaign we could measure only forces in longitudinal and vertical direction, in the DEM we obtain all 6 forces and torques.



(a) Measurement (red) and Simulation (blue) of excavator bucket in test rig for a *coarse sand* specimen. Left: The cutting depth is set to 0.15 m, the longitudinal dragging velocity to 0.12 m/s and the angle of incidence to 0°. Right: The cutting depth is set to 0.1 m, the longitudinal dragging velocity to 0.12 m/s and the angle of incidence to 25°. In gray are the unfiltered simulation results, the coloured graphs represent the fifth order butterworth filtered signal with cutoff frequency of 1 Hz



(b) Measurement of mean excavator bucket forces in bucket trench test rig. From left to right: Mean forces as a function of the cutting depth d and of the dragging velocity v

Figure 5.8: Measurement and simulation of coarse sand in bucket trench

Numerical Example: Excavator Bucket in Gravel testbed For a more realistic example, we consider the same testbed with an excavation tool. We investigate this scenario in an experiment at the soil laboratory of Technische Universität Kaiserslautern, see Figure 5.5 and Section 5.3. That is, we replace the plate by the model of a commercial excavator bucket with a cutting width of 0.305 m. The bucket has a total weight of 31.6 kg. This kind of excavator bucket is typically used during small-scale excavations with compact excavators. We chose the tool's cutting depth d between 0.05 and 0.25 m , while the velocity v in longitudinal x-direction lies between 0 and 1 m/s. Additionally, we change the angle θ around the cutting edge in the range of -30 to 30°. In this example, we choose a finer meshing of our parameter space, resulting in a total of 125 basis simulations. For simulation results, we refer to Section 5.3, where we compare the Lookup approach to the measurement and the full DEM simulation.

5.4 Interactive Soil Simulation on the Example of an Excavator

In this section, we describe the coupling of the soil-model, the visualization and the multibody framework of the excavator. We will therefore follow the steps described in Section 5.1. We discussed the soil parametrization and its validation in the preceeding two sections. For the application at the driving simulator, we choose the sand gravel mixture, see Section 5.2. Next, we need a useful model of an excavator. We describe a basic model and a more complex model in the succeeding subsection and give some insight into the classification of excavation trajectories. It is essential, to define coordinate systems in the MBS model and in the soil model, which have to coincide. Suitable offline simulations have to be specified, covering the most relevant trajectories of the working range of the excavator. Thereafter, we generate an online model using the acquired offline simulation data. Finally, the online model can run on the driving simulator.

5.4.1 Multibody Model of an Excavator

An excavator can be modeled as an MBS in chain structure. For the sake of completeness, we shortly describe the following parts, without going into too much details. The main constituents of the excavator are presented in Table 5.3. Instead of distinguishing boom and adjustable boom, there exist excavators with a one-piece boom.

Car	Boom
4 Wheels	Boom
Front Axle	Adjustable Boom
Rear Axle	Stick
Undercarriage	Quick Fit
Superstructrure	Yoke
Blade	Connecting Rod

Table 5.3: Excavator Parts modelled by the Multibody System.

Classification of Excavation Trajectories In the paper [Ben+16, Figure 3, p. 2], the authors describe the generalized trajectory of an excavator bucket as a combination of the following basic maneuvers. The *start* phase describes the bucket position above the soil, with a positive inclination angle. It moves transversely downward to cut into the soil with its teeth or cutting edge. The second *constant velocity* phase describes the movement in lateral direction with constant velocity and constant cutting depth. The third *lifting* phase is a rotating maneuver from a positive inclination angle to a negative inclination angle in order to accumulate the material inside the bucket. The fourth *transport* phase, moves the bucket without changing the angle of incidence, so that as little material as possible is lost. This might also comprise a change in the swing angle and the driving wit the excavator. The fifth *unloading* phase describes a rotation of the bucket from a negative to a positive angle of inclination emptying the accumulated material from the bucket. The sixth and final *travel to start* phase closes the circle to beginn with the next maneuver in the start phase.

The force prediction of the DEM RNN approach covers phase one to three. Here, the major forces arise, because the soil is broken and possible cohesive forces need to be overcome. The DEM LUT approach covers only phase two. Phases four to five can be simulated by adding a mass to the multibody modell of the excavator, thus the time consuming offline DEM simulation is not required. In phase six, the bucket is empty and no additional forces arise.

Multibody Excavator Model For the trajectory generation, we designed a reduced Simscape Multibody model of an excavator. A second more complex SimMechanics model with hydraulics is currently in use at RODOS. However, the latter model is slower due to its complexity and the manual trajectory generation is cumbersome.

We modeled the excavator, consisting of a seven bodies in Simscape Multibody.

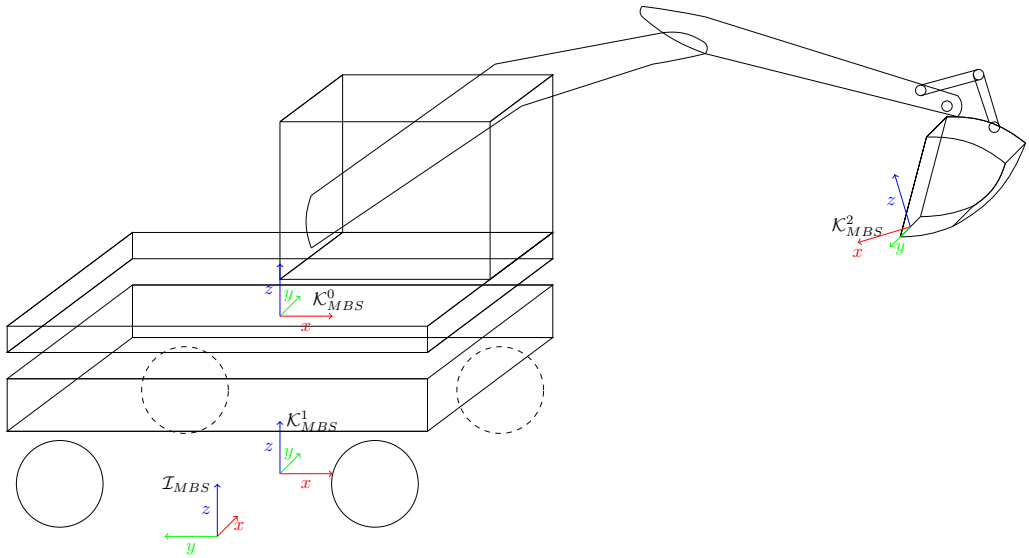


Figure 5.9: Overview on the relevant coordinate frames of the multibody model of an excavator for the coupling with soil models

The undercarriage is connected to the superstructure by a revolute joint, rotating around the global z-axis. Superstructure and boom, boom and stick, and stick and bucket are all connected by revolute joints all axis aligned. Thus, we have a swing angle rotating the car around the undercarriage and may move in a local x-z-plane in a coordinate frame of the bucket relative to the superstructure. We implemented two versions of this model. First, we can prescribe the four angles (swing, superstructure-boom, boom-stick, stick-bucket) and compute the resulting bucket trajectory. Second, we connect the bucket tip with the superstructure using a planar joint. A planar joint constrains the motion in a two dimensional way, leaving two translational and one rotational degree of freedom. Then, we can prescribe the position in the x-z-plane of the excavator bucket and obtain the resulting angles in the revolute joints. Therefore, we specify certain x-z-positions and the angle of incidence θ and interpolate the trajectory. This guarantees that we never leave the working space of the excavator.

Coordinate Systems In order to couple GRAPE or later on the hybrid models from Section 4.3.1 and 4.3.2 with the multibody model of the excavator, we need to make sure the coordinate systems coincide. Let us first look at the multibody model of the excavator and specify the coordinate frames as introduced in Section 2.5, see Figure 5.9. The transformations between the coordinate systems correspond to

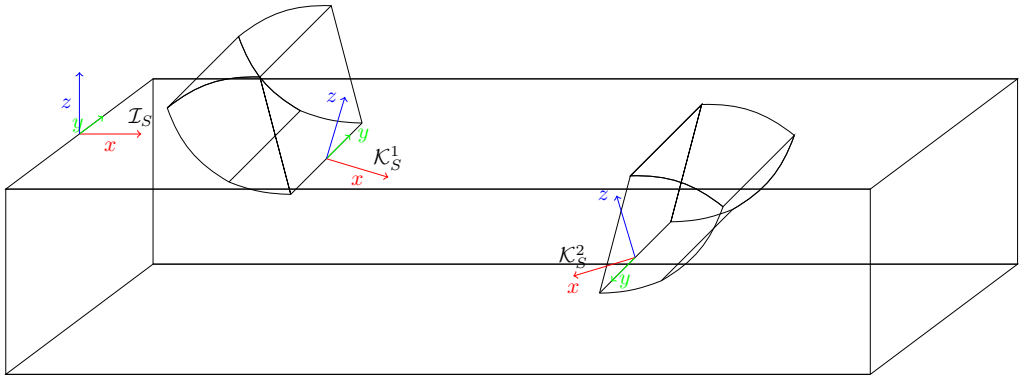


Figure 5.10: Overview on the relevant coordinate frames of the soil simulation for the coupling with the multibody model of an excavator

afine linear mappings. Here, we focus only on the rotation matrix. We consider a global coordinate frame $\mathcal{I}_{MBS}$. We choose a body fixed coordinate frame $\mathcal{K}^0_{MBS}$ at the super structure, with rotation matrix $\mathcal{R}_{\mathcal{K}^0_{MBS}\mathcal{I}}$. We consider this rotation matrix as a passive transformation from global coordinates $\mathcal{I}_{MBS}$ to $\mathcal{K}^0_{MBS}$. This rotation matrix mainly captures the swing angle around the global z-axis between undercarriage and superstructure. We define a second coordinate system just below $\mathcal{K}^0_{MBS}$ with global height $z = 0$ and name it $\mathcal{K}^1_{MBS}$. The rotation matrix

$$\mathcal{R}_{\mathcal{K}^0_{MBS}\mathcal{K}^1_{MBS}} = \mathcal{R}_{\mathcal{K}^0_{MBS}\mathcal{I}_{MBS}}\mathcal{R}^T_{\mathcal{K}^1_{MBS}\mathcal{I}_{MBS}} = \mathcal{R}_{\mathcal{K}^0_{MBS}\mathcal{I}_{MBS}}\mathcal{R}_{\mathcal{I}_{MBS}\mathcal{K}^1_{MBS}}$$

is constant in time. As a second rigid body within the MBS of the excavator, we consider the bucket with bucket tip frame $\mathcal{K}^2_{MBS}$ and the center of gravity frame $\mathcal{K}^3_{MBS}$. Again, the relative rotation $\mathcal{R}_{\mathcal{K}^2_{MBS}\mathcal{K}^3_{MBS}}$ is constant in time. The rotation matrix between superstructure and bucket tip $\mathcal{R}_{\mathcal{K}^1_{MBS}\mathcal{K}^2_{MBS}}$ corresponds to the bucket angle with respect to the local y-axis. We measure the time-dependent rotation matrix $\mathcal{R}_{\mathcal{K}^1_{MBS}\mathcal{K}^2_{MBS}}$ in order to compute the angle θ around the y-axis. The matrix $\mathcal{R}_{\mathcal{K}^1_{MBS}\mathcal{K}^2_{MBS}}$ can be interpreted as an active rotation $\mathcal{K}^1_{MBS} \to \mathcal{K}^2_{MBS}$.

Second, let us specify the relevant coordinate systems within the soil model. For the soil modeled in GRAPE, we look at a global coordinate frame $\mathcal{I}_S = \mathcal{K}^0_S$ and a body-fixed frame at the bucket tip $\mathcal{K}^1_S$ or $\mathcal{K}^2_S$, compare Figure 5.10. We need to ensure that this corresponds to $\mathcal{K}^2_{MBS}$. The rotation matrix $\mathcal{R}_{\mathcal{K}^2_S\mathcal{I}_S}$ describes the rotation between local body fixed coordinate system and world. Difficulties arise when we collect data with a bucket orientation different from the one of the real time simulation $\mathcal{K}^2_S$. Then, we need to rotate with respect to the systems $\mathcal{K}^1_S$ and $\mathcal{K}^2_S$. This active transformation corresponds to a rotation $\mathcal{R}_z(\pi)_{\mathcal{I}_S\mathcal{I}_S}$ around the

global z-axis. Then, we obtain

$$\mathcal{R}_{\mathcal{I}_S\mathcal{K}_S^1} = \mathcal{R}_z(\pi)_{\mathcal{I}_S\mathcal{I}_S}\mathcal{R}_{\mathcal{I}_S\mathcal{K}_S^2} = \mathcal{R}_z(\pi)_{\mathcal{I}_S\mathcal{I}_S}\mathcal{R}_{\mathcal{K}_{MBS}^1\mathcal{K}_{MBS}^2}. \tag{5.1}$$

Force-Displacement Coupling The MBS simulation prescribes a motion of boom, arm, quick fit and finally the bucket. As described in the previous paragraph, we couple the systems at the bucket tip point. The reason is, that the generation of Lookup Tables was issued with respect to the bucket tip coordinate system in order to capture soil contact more accurately. We capture the motion of the bucket in $\mathcal{K}_{MBS}^2$ with respect to $\mathcal{K}_{MBS}^1$ and translate it into GRAPE's coordinate system $\mathcal{K}_S^2$ with respect to $\mathcal{I}_S$. Forces and torques are computed in the global coordinate frame $\mathcal{I}_S$ acting at the bucket tip coordinate frame $\mathcal{K}_S^2$. If we have precomputed forces, e.g. using the DEM LUT approach, with respect to $\mathcal{K}_S^1$ we need to transform them analogously to Equation (5.1). With a slight abuse of notation, we write $\boldsymbol{F}_{\mathcal{K}_S^1}$ as the force in global coordinates acting at the origin of $\mathcal{K}_S^1$. That is, we take the forces $\boldsymbol{F}_{\mathcal{K}_S^1}$ and multiply with the transpose of $\mathcal{R}_z(\pi)_{\mathcal{I}_S\mathcal{I}_S}^T = \mathcal{R}_z(\pi)_{\mathcal{I}_S\mathcal{I}_S}$, as it is symmetric, i.e.

$$\boldsymbol{F}_{\mathcal{K}_S^2} = \mathcal{R}_z(\pi)_{\mathcal{I}_S\mathcal{I}_S}\boldsymbol{F}_{\mathcal{K}_S^1} \quad \text{and} \quad \boldsymbol{T}_{\mathcal{K}_S^2} = \mathcal{R}_z(\pi)_{\mathcal{I}_S\mathcal{I}_S}\boldsymbol{T}_{\mathcal{K}_S^1}.$$

We apply the forces and torques in the MBS at the point $\mathcal{K}_{MBS}^2$ and need to make sure, that we apply the forces locally. Although the force torque vector has been computed in global coordinates $\mathcal{I}_S$ at the point $\mathcal{K}_S^2$, in the MBS world, we operate in a local world between $\mathcal{K}_{MBS}^1$ and $\mathcal{K}_{MBS}^2$.

5.4.2 Selection of Appropriate Training Data

For the application of the online methods described in the previous chapter in Section 4.3, we need to perform offline simulations and acquire the necessary data.

Lookup Basis Simulations The basis simulations for the Lookup Table, are computed similar to the previous examples in Section 4.3.1 and Section 5.3. The third example of the generation of a Lookup Table aims at the application of the DEM LUT approach at the driving simulator RODOS. We use a bucket geometry of cutting width of 1.20 m. The bucket has a total weight of 570 kg and is typically used in medium scale excavators. We choose the buckets cutting depth d between 0.05 and 0.8 m. The minimal depth submerges the bucket fully in the particle bed. We subdivide the interval $[0.05, 0.8]$ in steps of 0.15 m, resulting in six different depth values. The velocity lies between 0.05 and 0.85 m/s. Again, the angle of

incidence θ is in between $-30°$ and $30°$ in the bucket coordinate system $\mathcal{K}_S^1$. In total we perform $6 \times 5 \times 5 = 150$ offline basis simulations. Due to our experience with the previous two examples, we chose lower longitudinal velocities and focus on the variation of the cutting depth d.

The basis simulations has coorcinates in $\mathcal{K}_S^1$, but the maneuvers in and the online phase use coordinates in $\mathcal{K}_S^2$. This means that the trained net does not recognize the data from basis simulations as fitting for the online simulation. Thus, we oriented the data from the basis simulations as follows.

Oriented Basis Simulations As discussed in Paragraph 5.4.1, it makes sense, to rotate the data obtained in $\mathcal{K}_S^1$, to fit into $\mathcal{K}_S^2$. That is, we have the rotation matrix $\mathcal{R}_{\mathcal{K}_S^1 \mathcal{I}_S}$, which we multiply with $\mathcal{R}_z(\pi)_{I_S I_S}$ in order to orient it with respect to $\mathcal{K}_S^2$.

$$\mathcal{R}_{\mathcal{I}_S \mathcal{K}_S^2} = \mathcal{R}_z(\pi)_{I_S I_S} \mathcal{R}_{I_S \mathcal{K}_S^1}. \tag{5.2}$$

We use this data in the application of Lookup Tables as describes in 4.3.1. For artificial simulations close to the basis simulations, when the bucket was dragged through soil with little variation of the Lookup parameters, the results looked promising, see Figure 5.11. However, when the method is used with real excavation maneuvers, the DEM LUT approach underestimates forces and torques, above all the vertical forces in z-direction.

MBS Excavation Trajectories To achieve better accuracy, we use data generated with the MBS model of the excavator. Defining trajectories by interpolating predefined points lead to smooth curves and it is possible to set desired cutting depths and angles of incidences. However, the collected data do not result in RNNs with satisfying approximation quality, because the characteristic of the input of the driving simulator differs. Therefore, we tried to get as close to the online trajectories as possible, see the succeeding paragraph.

RODOS Excavation Trajectories In addition to the artificially defined trajectories as described in the last paragraph, we also collected data by operating in the cabin of an excavator and perform realistic digging maneuvers, see Figure 5.15. Here, we modelled the soil impact by an elastic damper, forbidding low cutting depths. We operated the cabin inputs for the trajectory generation slowly to obtain only subtle oscillations. During the excavation, we recorded the angles between

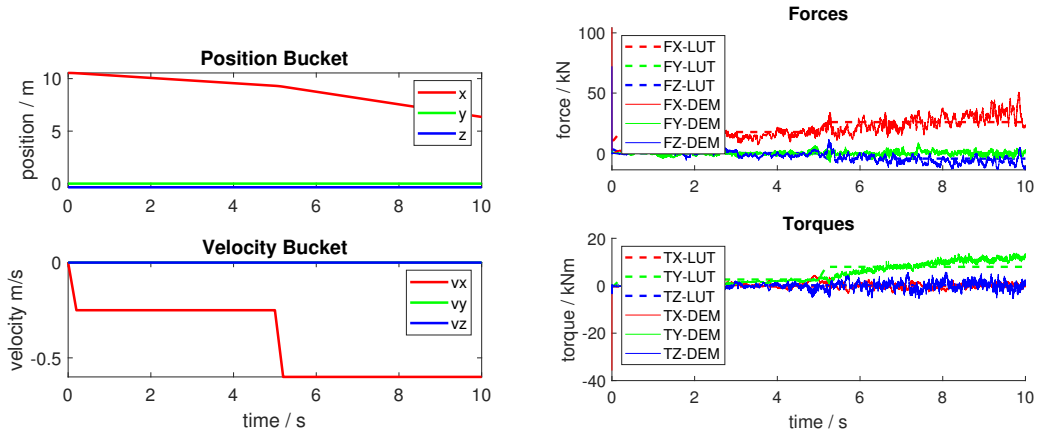


Figure 5.11: Trajectory and force torque output of LUT compared to full DEM simulation of a maneuver similar to the LUT basis simulations

the multibodies of the excavator model. Subsequently, in a postprocessing step, we translated the measured angles between the bodies of the excavator MBS into a bucket tip trajectory. We extracted the relevant parts of the trajectories, covering phase 1 (start), phase 2 (constant velocity) and phase 3 (lift), compare Paragraph 5.4.1. These parts served as input for DEM offline simulations. When the cutting depth seemed too deep, to obtain a realistic dig, we added an offset, in order to obtain forces below 500 kilonewtons. Otherwise, especially the forces in vertical z-direction become unrealistically large, which would result in instable machine response. Thereafter, the trajectories serve as input for offline DEM simulations. We associate the acquired forces to the mapping

$$f : \boldsymbol{u} \longmapsto f(\boldsymbol{u}) := \boldsymbol{y}, \text{ with } \boldsymbol{u} = (\boldsymbol{x}^T, \boldsymbol{v}^T, \boldsymbol{q}^T, \boldsymbol{\omega}^T)^T \text{ and } \boldsymbol{y} = (\boldsymbol{F}^T, \boldsymbol{T}^T)^T.$$

The training data for the RNNs in the upcoming examples comprises 96 simulations of five to fifteen seconds. In total, we generated almost two hours of simulated excavation cycles. The data has the form $(\boldsymbol{u}, \boldsymbol{y})$, where $\boldsymbol{u}$ describes the input trajectory and $\boldsymbol{y}$ the forces and torques. Prior to training, we downsampled the data to 10 Hz. Afterwards, we append the trajectories to obtain one large sequence of input vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ and their respective outputs $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$.

5.4.3 Coordinate Calibration

In order to validate the calibration of the coordinate transforms, we perform a simulation similar to the basis simulations for the DEM LUT approach, but oriented

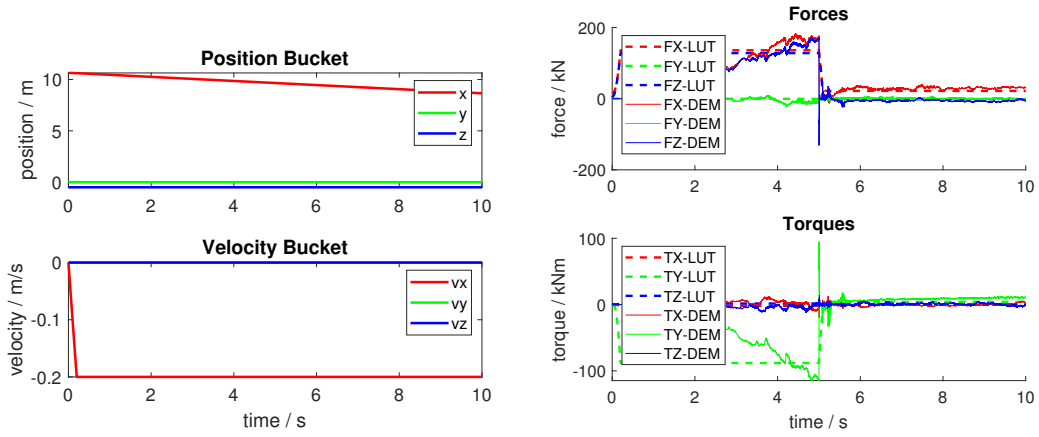


Figure 5.12: Trajectory (left) and force torque output of LUT compared to full DEM simulation of a maneuver similar to the LUT basis simulations

in the opposite direction. That is, we are in the coordinate frame $\mathcal{K}_S^2$, but in the basis simulations for the DEM LUT approach, we use data recorded in $\mathcal{K}_S^1$. Within the simulation, we consider a bucket moving in negative x-direction with moderate speed between $v_x = -0.25$ m/s and $v_x = -0.6$ m/s. We chose the angle around the y-axis to be zero and change the cutting depth d from $d_0 = -0.35$ m to $d_1 = -0.65$ m. The trajectory and the force results are presented in Figure 5.11. The forces agree, although the linearly increasing trend in the second half of the DEM simulation longitudinal forces FX cannot be captured by the LUT.

We choose a second maneuver where the bucket changes from an open orientation angle of $-30°$ to a closing angle of $30°$, see Figure 5.12. The parameter change in the DEM simulation is nonsmooth which leads to abrupt acceleration forces. In the first half of the simulation, the LUT first overestimates and then underestimates the DEM forces. Again the linear increase in the longitudinal force FX is not captured. All in all, the LUT force prediction captures the main trend of the DEM simulation forces.

Online Model Generation We chose the Lookup 1 approach, see Variant 1 in Section 4.3.1. This method yields almost smooth force trajectories. We load the data in the MATLAB workspace and access it via a MATLAB/Simulink block. For the architecture of the RNN, we choose a net with one hidden layer and five recurrent layers. A network with two or more hidden layers drastically increases the degrees of freedom and thus the training time. Initially, the results with more than one hidden layer demonstrated no advantageous approximation behavior, so

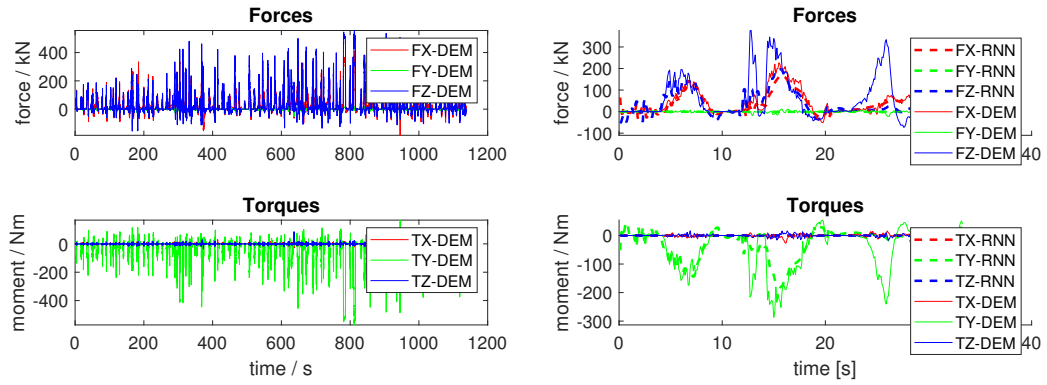


Figure 5.13: RODOS excavation trajectories: Training Data and test data for RNN compared to DEM simulation.

we tried to keep the model as simple as possible. We compare to RNNs with the following structure.

On the one hand, we compare the RNN with 7 relevant inputs, containing $\boldsymbol{x}_x$, and $\boldsymbol{x}_z$, two non-zero entries in the quaternion $\boldsymbol{p}$, the velocities $\boldsymbol{v}_x$ and $\boldsymbol{v}_z$ and the angular velocity $\boldsymbol{\omega}_y$ in the upcoming numerical study. We applied the Scaling Variant 1 subtracting the temporal mean value of one input signal calculated over all training data and dividing by the standard deviation, see Subsection 2.4.3.

On the other hand, we use the full trajectory incorporating also close to zero entries in the net, resulting in 13 inputs. Therefore, we applied the Scaling Variant 2, subtracting in each time step the mean over all input signals and dividing by the standard deviation of this time step.

Both nets have 5 recurrent layers. We incorporate the trained nets into a MATLAB/Simulink block and name the first RNN7 and the second RNN13.

Deployment at RODOS We deployed the designed online applications discussed in the preceding sections, at the driving simulator RODOS. Here, we will briefly describe the design of the simulator and point out the commercial software used for the deployment of the interactive soil applications. A KUKA robot arm forms the basis of the driving simulator and its development is described in [Kle15a], see also [KUK]. Currently, three types of vehicles are operational, namely a passenger car, an agricultural tractor and an excavator. A full size passenger cabin of each of the three vehicles is mounted on top of the robot arm. A vehicle model can be developed and run on a realtime capable computer. Currently, a SCALEXIO by the electronic

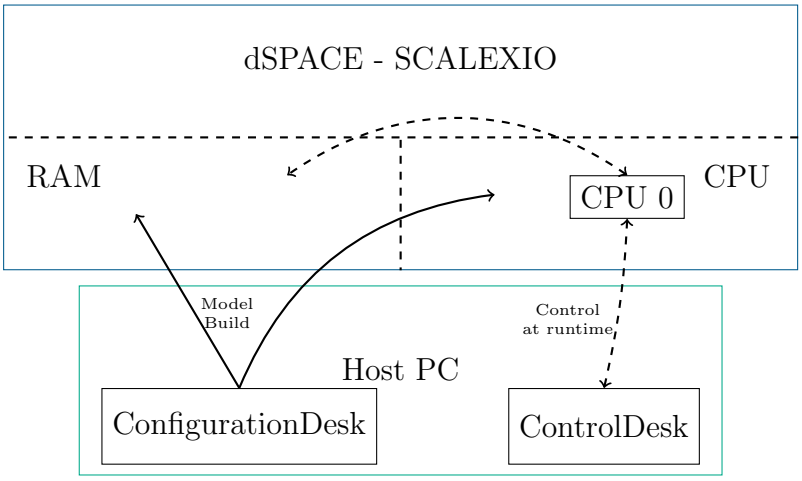


Figure 5.14: Sketch of the dSPACE environment for the realtime model.

control unit developer dSPACE is in use as realtime system, see [dSPb; dSPa]. The preparation and runtime control of the application running on the realtime platform uses two software frameworks supplied by dSPACE. These software components run on a separate host computer. These are ConfigurationDesk and ControlDesk. ConfigurationDesk acts as a comfortable wrapper for the vehicle model which also serves as a compiler. The compiled application starts the vehicle model on the SCALEXIO. The memory designation on the realtime system is also part of the building process and is static during runtime. ControlDesk on the other hand interacts with the realtime application at runtime, reads and writes the RAM of the SCALEXIO, thereby making possible the adaption of model parameters. The general structure is illustrated in Figure 5.14. The online models described in Section 4.3.1 and 4.3.2 are modeled as Simulink blocks in MATLAB Simulink and appended to the excavator model. Then, the application is compiled and imported via ConfigurationDesk.

We extend the vehicle model of the excavator (currently a VOLVO EW160E) with the soil-tool interaction block. The vehicle model communicates the bucket tip dynamics to the online soil model and this block calculates and returns the approximated forces and torques to the vehicle model.

Figure 5.15: From left to right: Unity visualization, simulator setup for trajectory generation, and mounted excavator cabin in RODOS

5.5 Applications of LUT and RNN - Numerical Results

In Figure 5.16, we demonstrate the trajectory of a real excavation maneuver, and the force response of LUT and DEM simulation. The oscillations in the velocities arise due to the vibrating multibody model of the excavator and due to the operator. A skilled excavation operator might reduce the amount of oscillations in the velocities and reduce the vertical forces. The forces magnitude coincides initially, however, only parts of the maneuver are predicted correctly by the LUT approach. When the bucket vertical position is positive, we obtain zero forces using the DEM LUT approach.

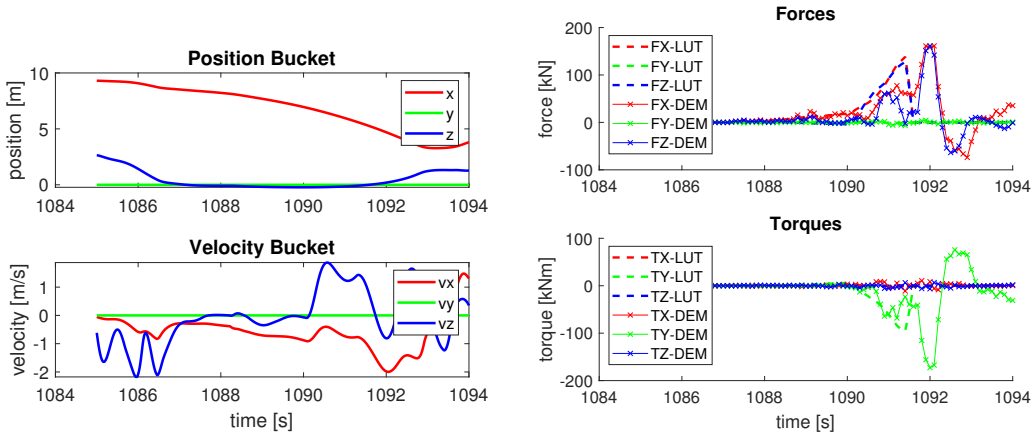


Figure 5.16: Trajectory and force torque output of LUT compared to full DEM simulation of a real excavation performed using a driving simulator.

Discussion of Lookup Table approximation The Lookup Table is sufficiently accurate to predict forces FX in global longitudinal direction, as long as the absolute value of the cutting depth is large enough, see Figure 5.18c. However, using our Lookup method with just three parameters d, θ, v_x is not sufficient to capture every aspect of a complex excavation maneuver. When the cutting depth is very low, the LUT does not respond with sufficiently large forces, see Figure 5.20c. The vertical forces are often underestimated, see Figure 5.22c. Furthermore, LUT does not predict local minima and maxima correctly, see Figure 5.20c. If the cutting depth is greater than zero, i.e. the bucket tip is above the soil surface, the forces calculated by LUT are zero, too. Thus, for maneuvers with accumulated surcharge material in front of the bucket and small absolute value of the cutting depth, the LUT does not correctly predict the lifting phase, compare to Section 5.4.

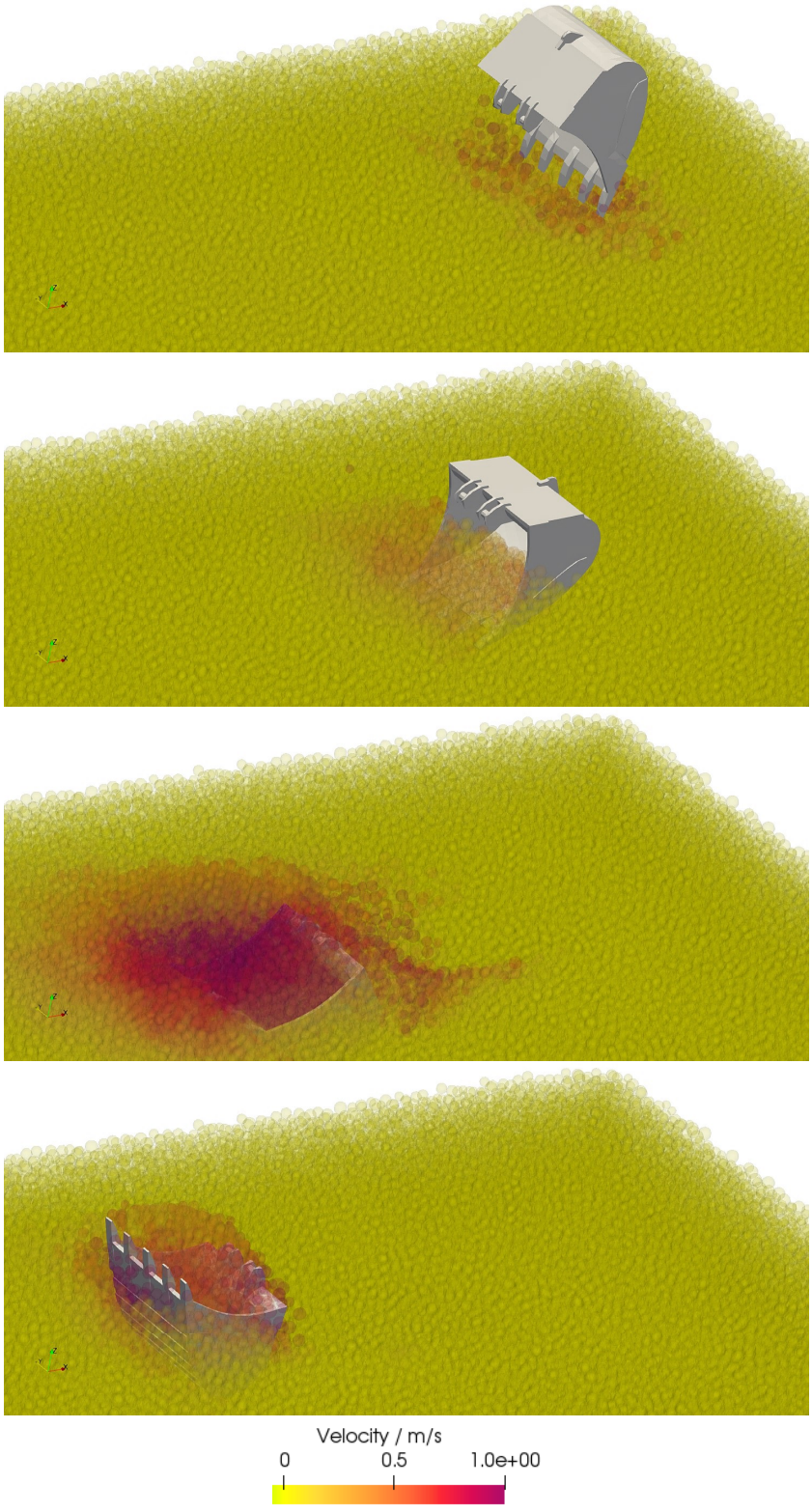
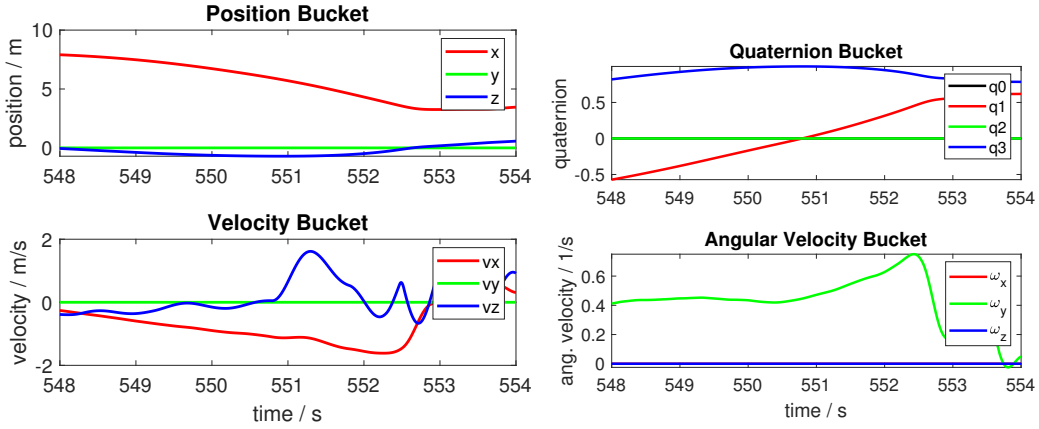
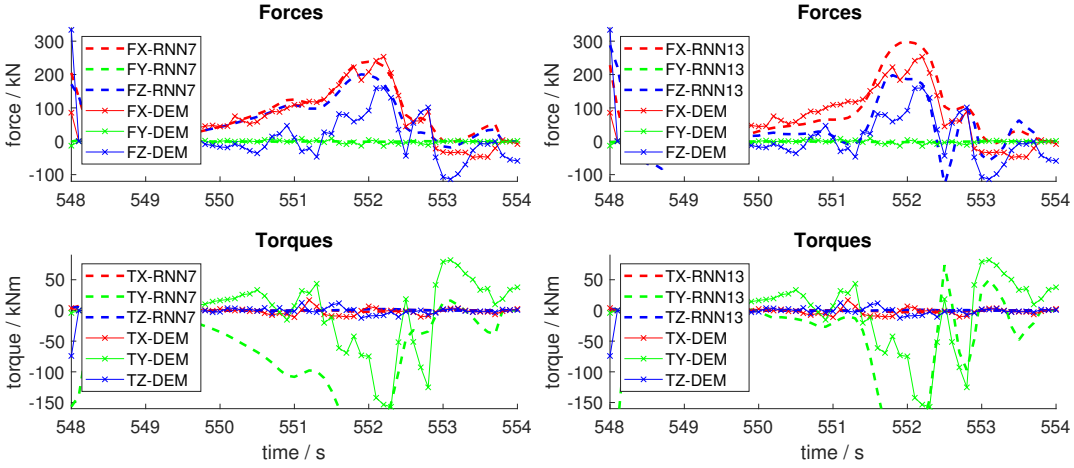


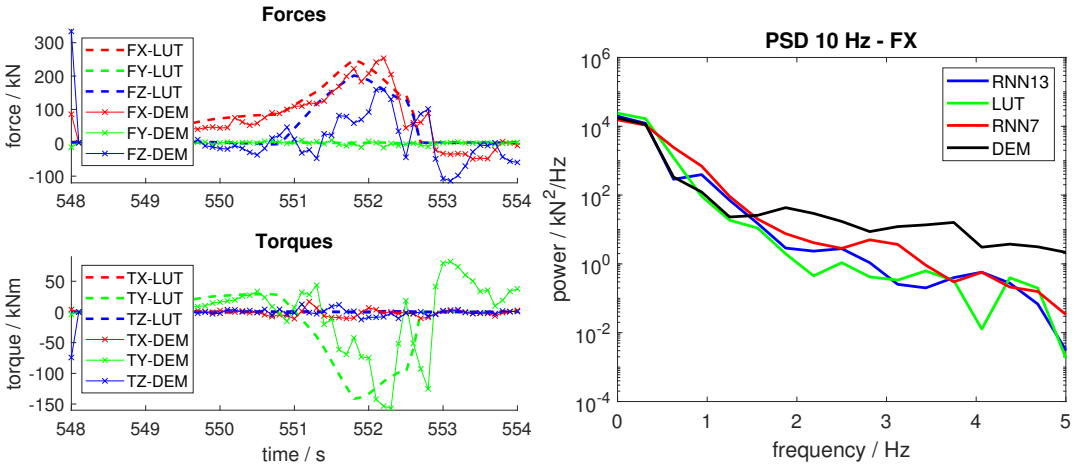
Figure 5.17: Magnitude of the particle velocity for Excavation Maneuver 1 for times $t = 548, 550, 552$, and 554 seconds, red color indicates high velocities



(a) Position and velocity (left), quaternion and angular velocity (right)

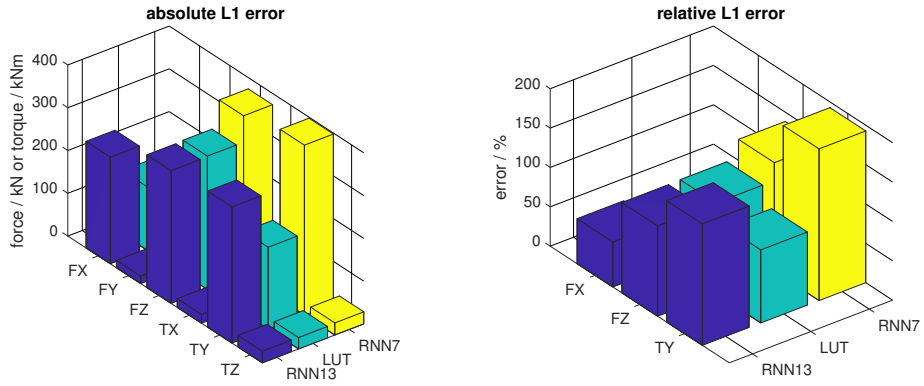


(b) Force torque output on an excavation maneuver of RNN with 7 inputs (left) and with 13 inputs (right), each compared to full DEM simulation of a maneuver

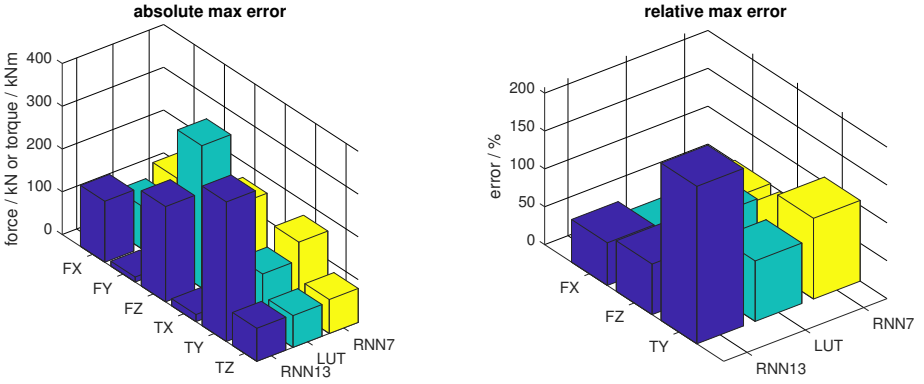


(c) Force torque output on an excavation maneuver of LUT (left), compared to full DEM simulation of a maneuver. The PSD of the different force time series (right)

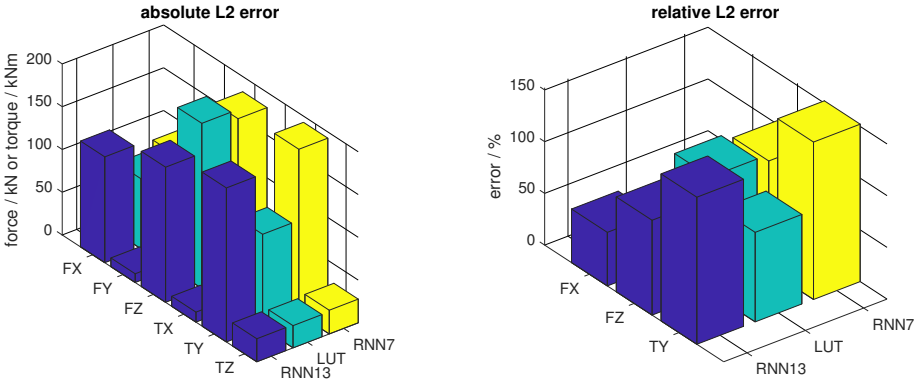
Figure 5.18: Excavation Maneuver 1



(a) Absolute and relative $\mathcal{L}^1$-error



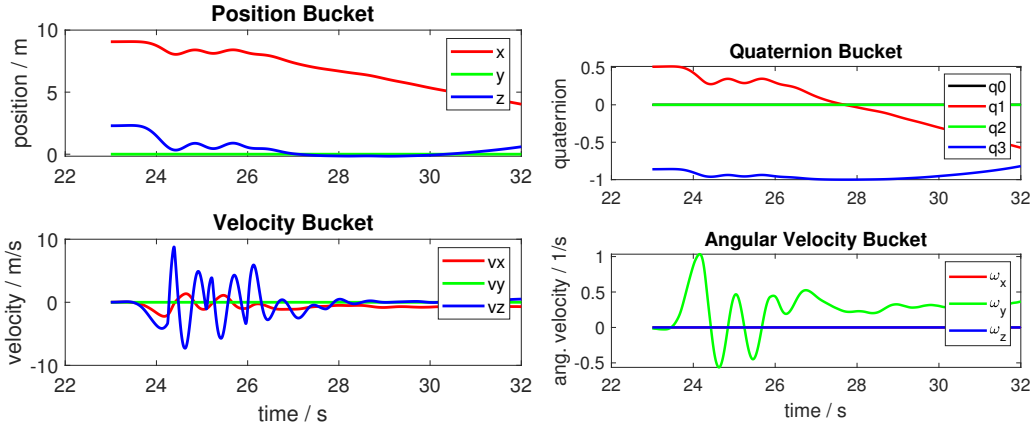
(b) Absolute and relative maximum error



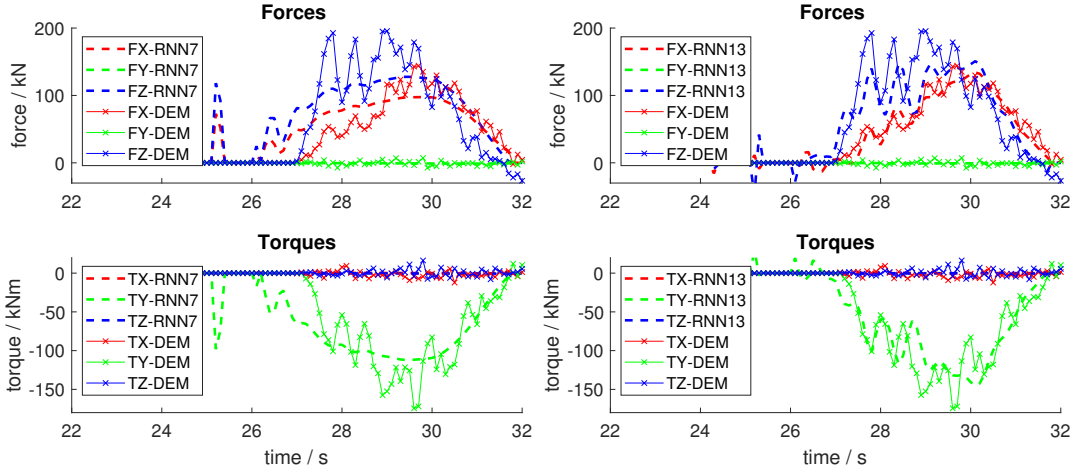
(c) Absolute and relative $\mathcal{L}^2$-error

Figure 5.19: Complete error Analysis of Excavation Maneuver 1

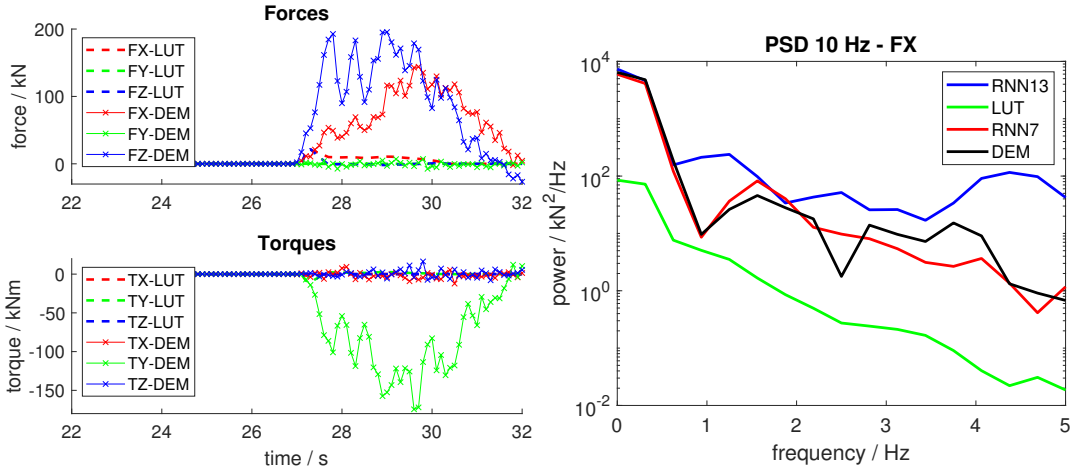
Discussion of Maneuver 1 We illustrate the first maneuver in Figure 5.17. Here, the colors indicate the particle velocity, yellow corresponds to zero velocities, while red indicates particle velocities above 1 meter per second. We evaluate the error measures defined in Section 2.6 and illustrate the results in Figure 5.19. The errors indicates a relative $\mathcal{L}^1$ error of up to 200 percent. All three methods correctly predict the non-zero forces at the start of the simulation, where the bucket tip, slightly penetrates the soil. Except for the vertical forces FZ, where the LUT outperforms RNN, the error of LUT and RNN are comparable. RNN7 has lower absolute errors in the component FX, but larger $\mathcal{L}^1$ and $\mathcal{L}^2$ errors regarding FZ and TY. In Figure 5.18b, RNN7 overestimated forces FZ and torques TY. Except for FZ, LUT outperforms the RNNs. The analysis of the frequency spectrum with a PSD plot indicates, that all methods reproduce the DEM frequencies accurately, see Figure 5.18c.



(a) Position and velocity (left), quaternion and angular velocity (right)

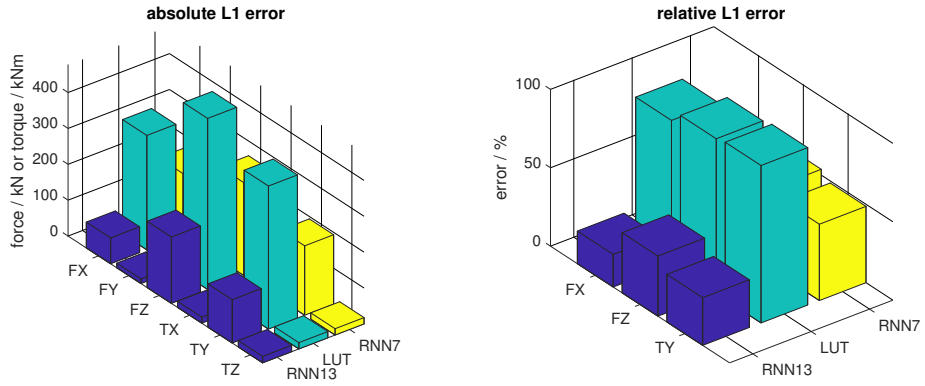


(b) Force torque output on an excavation maneuver of RNN with 7 inputs (left) and with 13 inputs (right), each compared to full DEM simulation of a maneuver

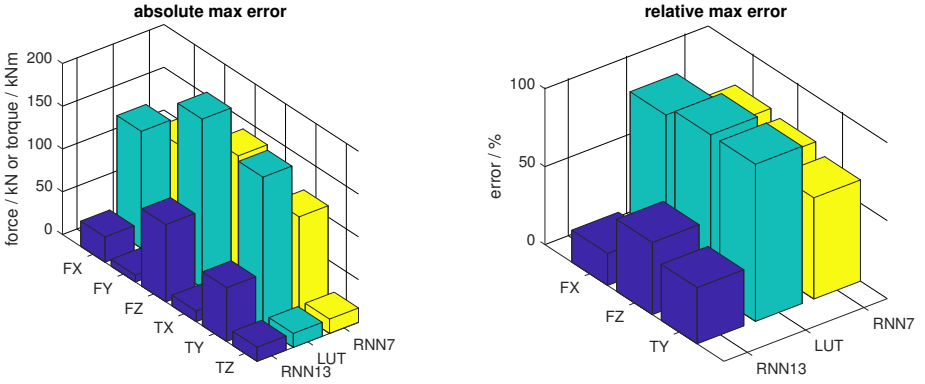


(c) Force torque output on an excavation maneuver of LUT (left), compared to full DEM simulation of a maneuver. The PSD of the different force time series (right)

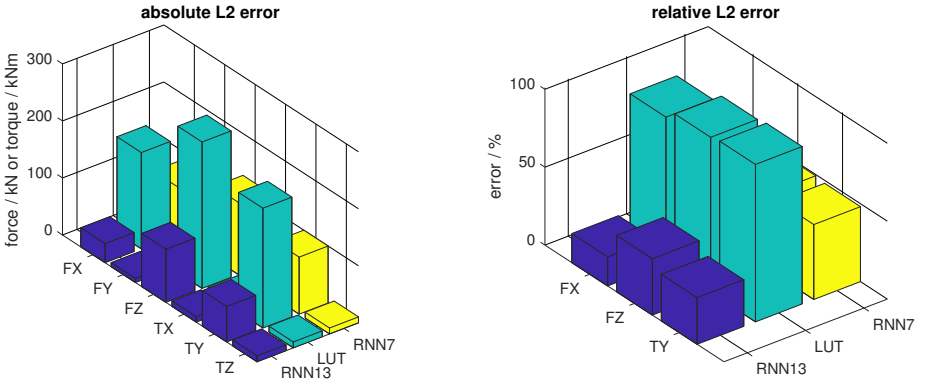
Figure 5.20: Excavation Maneuver 2



(a) Absolute and relative $\mathcal{L}^1$-error



(b) Absolute and relative maximum error



(c) Absolute and relative $\mathcal{L}^2$-error

Figure 5.21: Complete error Analysis of Excavation Maneuver 2

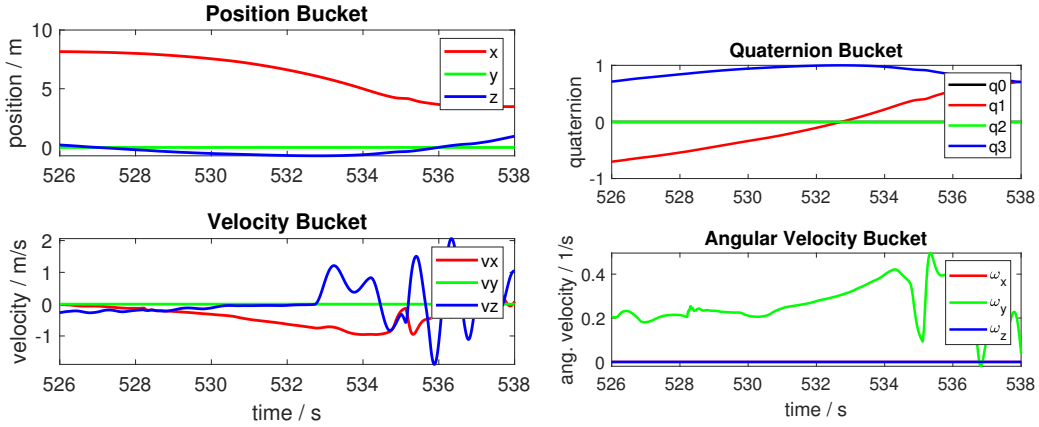
Discussion of Maneuver 2 The second maneuver reveals the deficiencies of the DEM LUT approach. The cutting depth remains below 0.2 m, thus the basis maneuvers accessed by the LUT yield only small reaction forces. The training data of RNNs comprise similar time series with low cutting depths and high forces, thus the approximation behavior is superior. The missing forces in the LUT approximation also influence the PSD, see Figure 5.20c.

The RNN7 reproduces the main behavior of the DEM solution. However, local minima and maxima are not approximated with all detail. RNN13 is superior here, especially regarding forces FZ. If we look at the absolute errors in Figure 5.21, we observe that RNN13 has lower errors compared to RNN7 and LUT. Also $\mathcal{L}^1$ and $\mathcal{L}_2$ errors of RNN13 are smaller, compared to RNN7 and LUT. The relative error of RNN13 is small, compared to RNN7 and LUT and is in the range of 10 to 30 percent. The LUT error is in the range of 100 percent, which we explain as follows. The LUT force prediction $\tilde{y}$ is small (close to zero), the relative error in all three norms reduces to

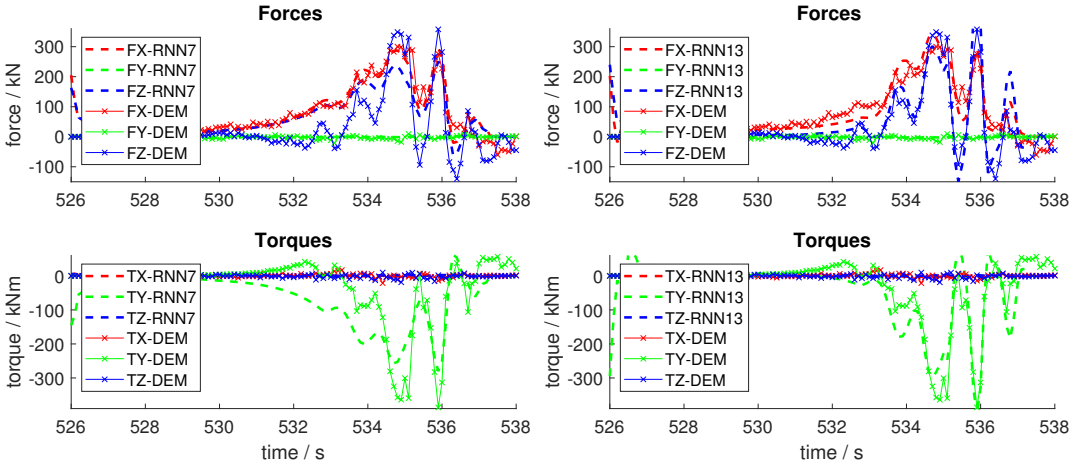
$$\frac{\varepsilon_k^a(y,\tilde{y})}{\varepsilon_k^a(y,0)} \approx \frac{\varepsilon_k^a(y,0)}{\varepsilon_k^a(y,0)} = 1,$$

where k refers to one of the defined error measures $(\infty, \mathcal{L}^1, \mathcal{L}^2)$, compare Section 2.6. Consequently, all relative LUT error measures for maneuver 2 are close to 100 percent.

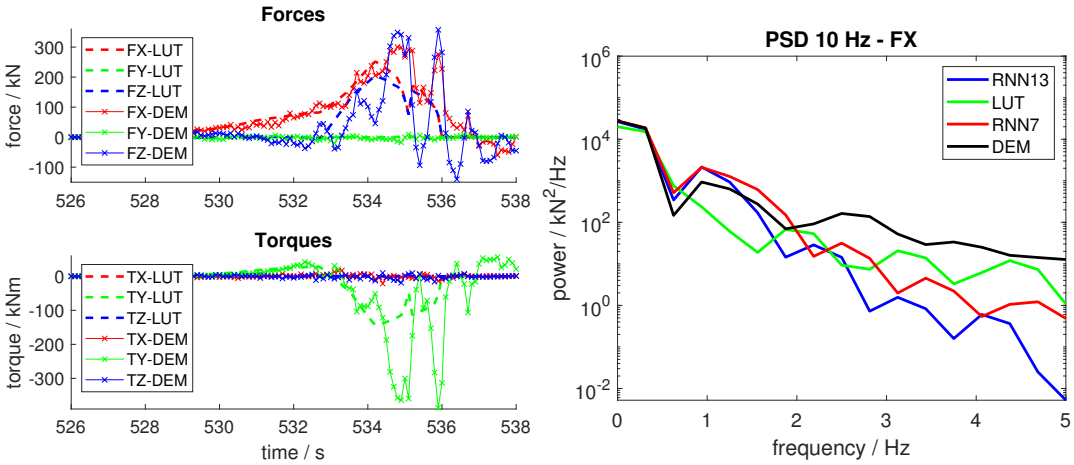
Discussion of Maneuver 3 This maneuver contains highly oscillating forces FZ. Similar to the previous maneuver, the LUT does not cover local maxima and minima. The network RNN7 with reduced input does cover the main behavior regarding oscillations, but avoids fully attaining minima and maxima, e.g. in FZ, compare to Figure 5.22b. The RNN13 with full input and different scaling approximates the oscillating behavior with better accuracy. This can be observed in the error plots, see Figure 5.23a and Figure 5.23c. However, the maximum error with respect to force FZ and torque TY is smaller for RNN7 compared to LUT and RNN13. The relative errors lie in the range of 30 to 100 percent.



(a) Position and velocity (left), quaternion and angular velocity (right)

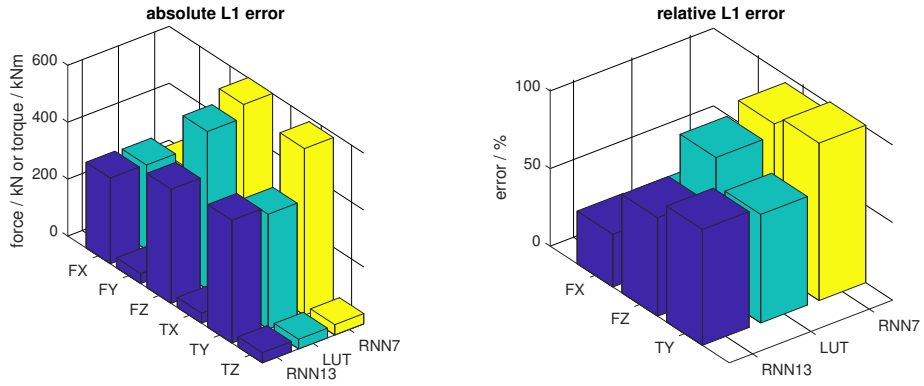


(b) Force torque output on an excavation maneuver of RNN with 7 inputs (left) and with 13 inputs (right), each compared to full DEM simulation of a maneuver

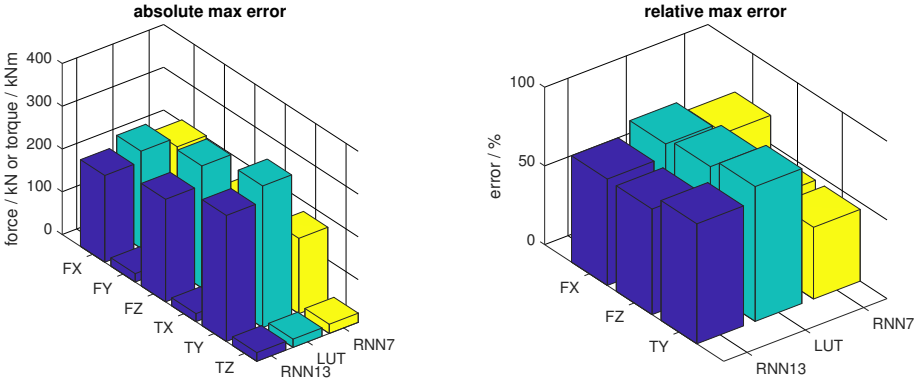


(c) Force torque output on an excavation maneuver of LUT (left), compared to full DEM simulation of a maneuver. The PSD of the different force time series (right)

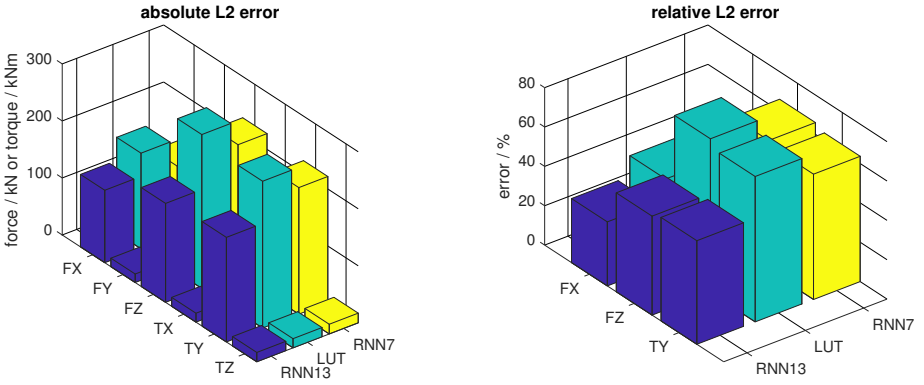
Figure 5.22: Excavation Maneuver 3



(a) Absolute and relative $\mathcal{L}^1$-error



(b) Absolute and relative maximum error



(c) Absolute and relative $\mathcal{L}^2$-error

Figure 5.23: Complete error Analysis of Excavation Maneuver 3

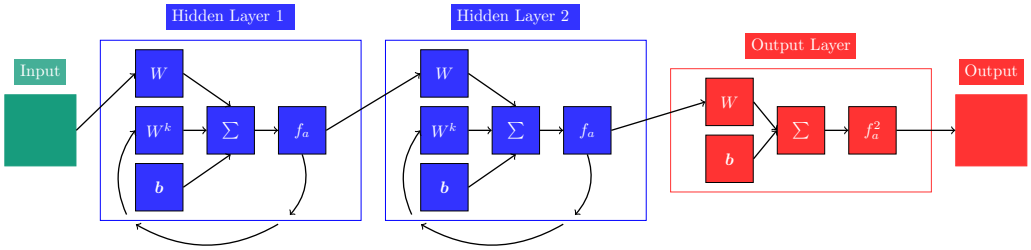
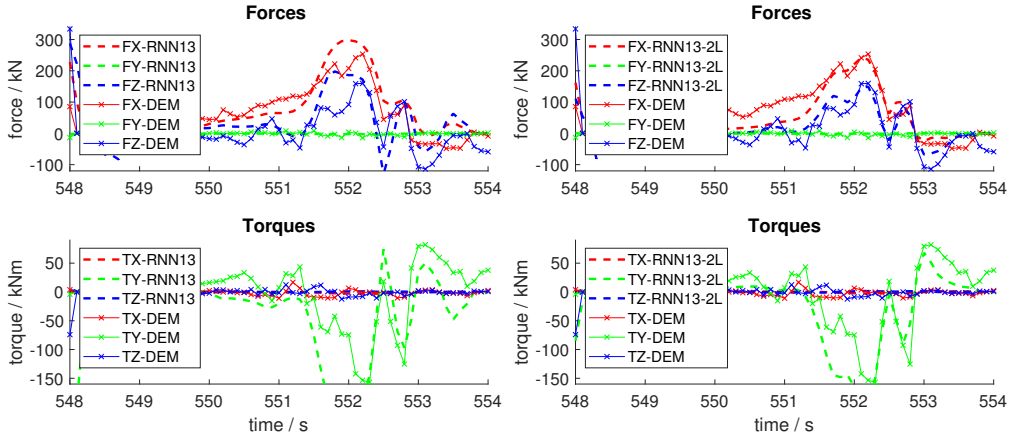


Figure 5.24: RNN with two hidden layers

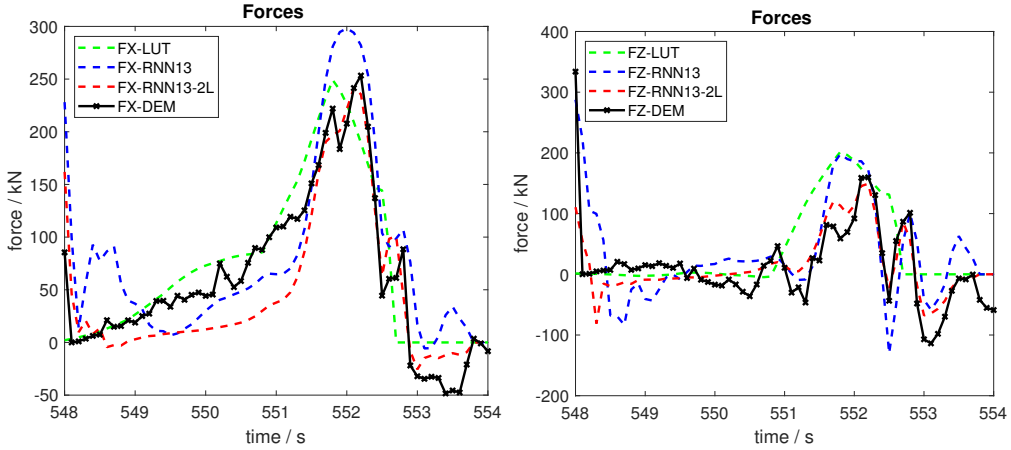
Summary of the Discussion All three methods succeed in approximating parts of the presented maneuvers. Only LUT fails for maneuver 2, where the absolute value of the cutting depth is small. For maneuver 2, RNN13 outperforms RNN7. In maneuver 1 and maneuver 3, we have no clear favorite method. RNN7 covers the main drift, while RNN13 catches local minima and maxima. To conclude, we prefer RNN13 and continue our study with it.

RNN with Two Hidden Layers In a second study, we used an RNN with 2 hidden layers and only 3 recurrent layers per hidden layer, instead of 5 for RNN13 and RNN7. We refer to the newly defined net by RNN13-2L. We simulate the same excavation maneuvers with the both RNNs, see Figure 5.25, Figure 5.26 and Figure 5.27.

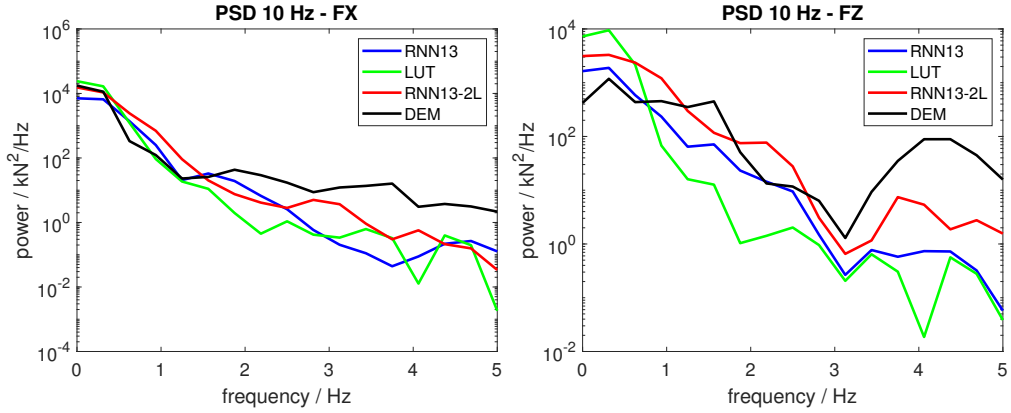
Discussion of the RNN with two Hidden Layers In Figure 5.27c on the right, RNN13 and RNN13-2L both approximate the frequencies satisfactorily. RNN13 is close to DEM between 0 and 2 Hz, the two-layered net RNN13-2L between 2 and 3 Hz. In Figure 5.25b on the left, RNN13 overestimates the longitudinal forces FX. Similarly, in Figure 5.27b on the right, RNN13 overshoots the local maxima in FZ. In Figure 5.26b, it is difficult to make out a difference of RNN13 and RNN13-2L.



(a) Force torque output on an excavation maneuver of RNN with 13 inputs and one hidden layer (left) and with 13 inputs and two hidden layers (right), each compared to full DEM simulation of a maneuver

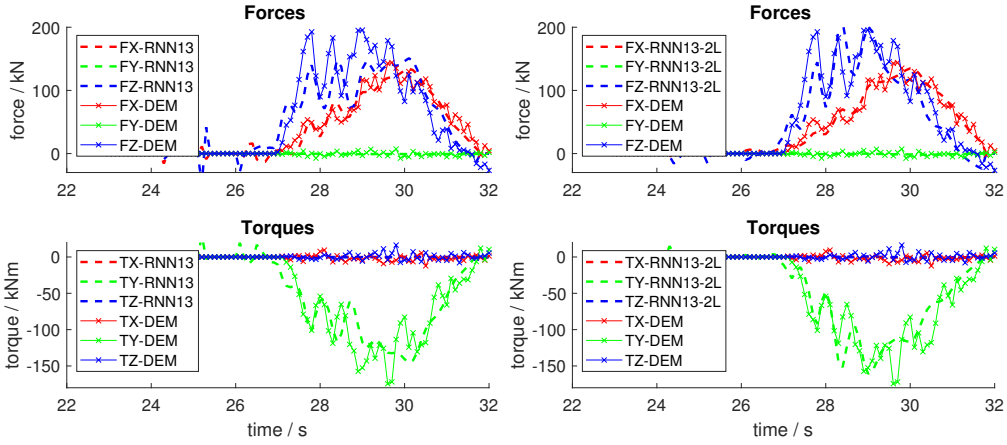


(b) Forces FX (left) and FZ (right), simulated with LUT, RNN13 and RNN13-2L compared to DEM

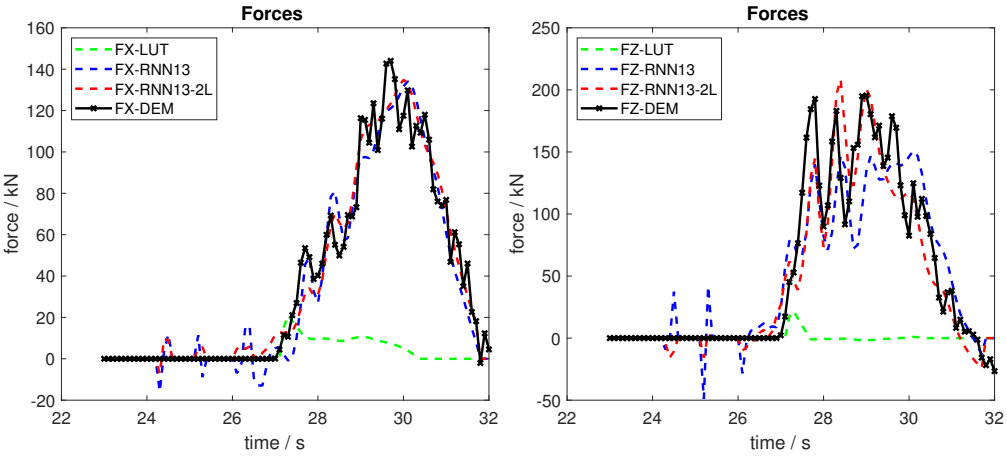


(c) PSD of forces FX (left) and FZ (right), simulated with LUT, RNN13 and RNN13-2L compared to DEM

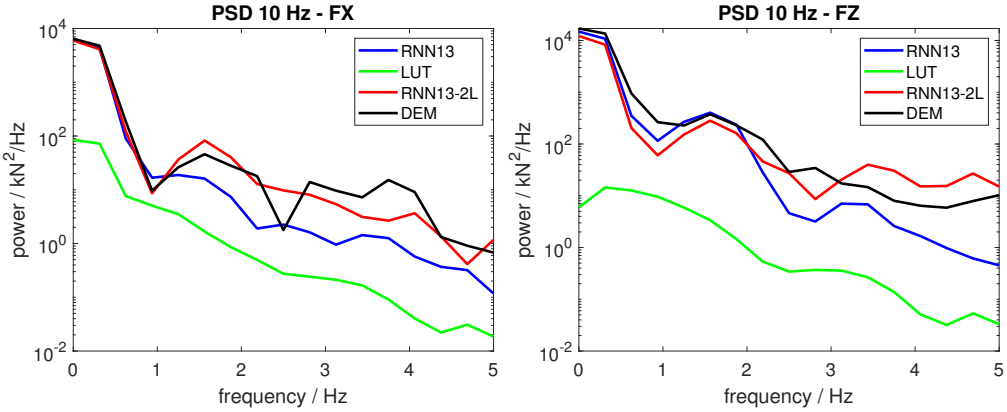
Figure 5.25: Maneuver 1 - comparison: RNN with two hidden layers



(a) Force torque output on an excavation maneuver of RNN with 13 inputs and one hidden layer (left) and with 13 inputs and two hidden layers (right), each compared to full DEM simulation of a maneuver

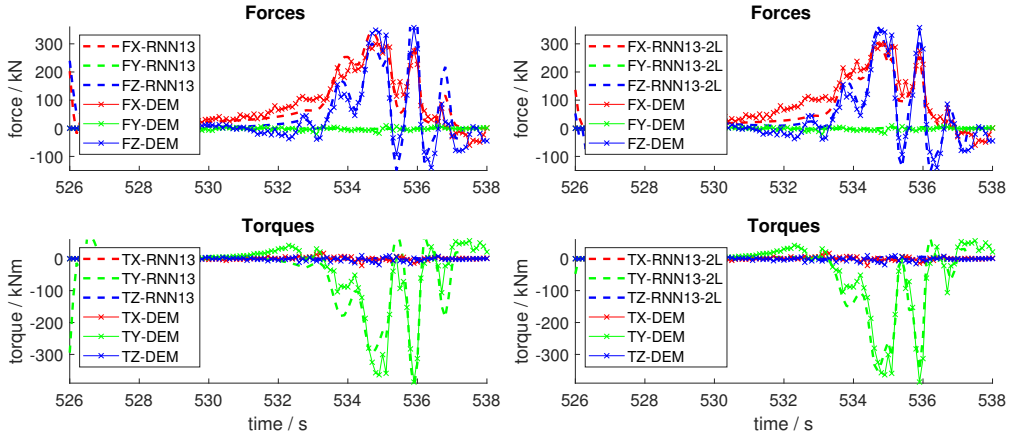


(b) Forces FX (left) and FZ (right), simulated with LUT, RNN13 and RNN13-2L compared to DEM

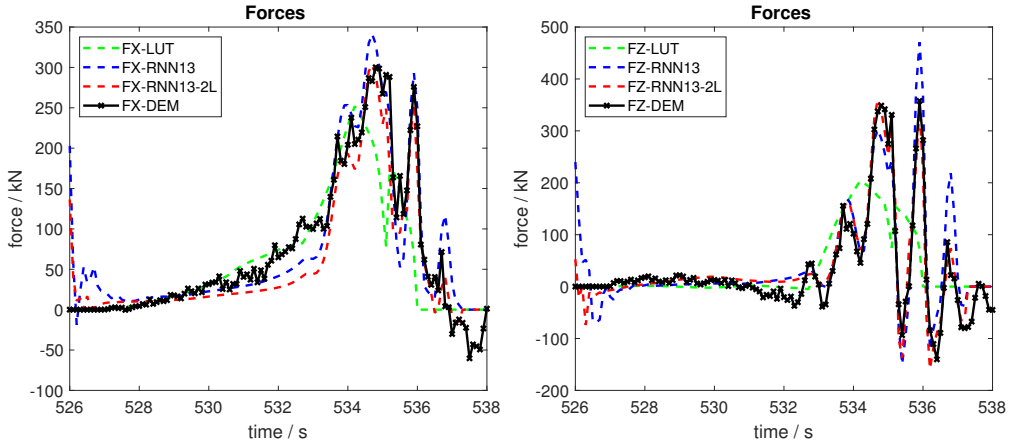


(c) PSD of forces FX (left) and FZ (right), simulated with LUT, RNN13 and RNN13-2L compared to DEM

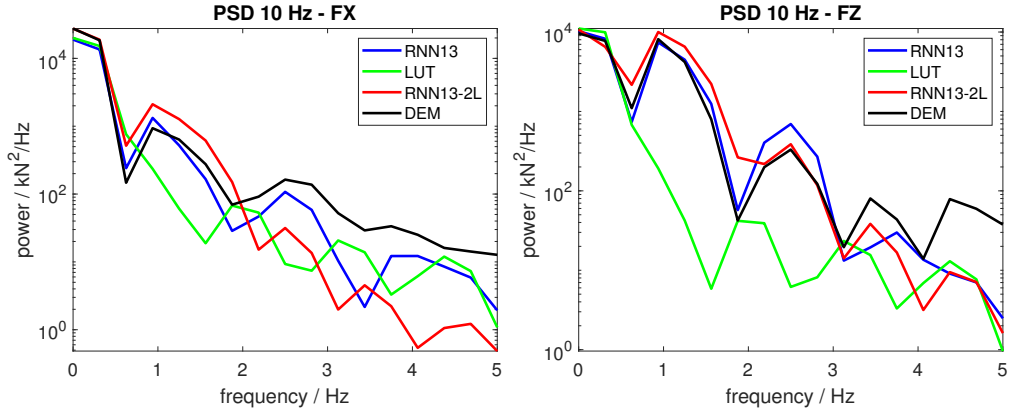
Figure 5.26: Maneuver 2 - comparison: RNN with two hidden layers



(a) Force torque output on an excavation maneuver of RNN with 13 inputs and one hidden layer (left) and with 13 inputs and two hidden layers (right), each compared to full DEM simulation of a maneuver



(b) Forces FX (left) and FZ (right), simulated with LUT, RNN13 and RNN13-2L compared to DEM



(c) PSD of forces FX (left) and FZ (right), simulated with LUT, RNN13 and RNN13-2L compared to DEM

Figure 5.27: Maneuver 3 - comparison: RNN with two hidden layers

Data Structure, Model Complexity and Computational Efficiency The LUT is generated using 150 basis simulations of 10 seconds simulation time each. We store them in form of a mean force and mean torque value for each coordinate axis. This results in a total of 900 doubles, we need to store for the model. The RNNs presented here, are based on training data consisting of 96 maneuvers of different length and almost 1200 seconds simulation time, see Figure 5.13. The RNN13 contains one hidden layer with 10 neurons and 5 recurrent layers. With 13 inputs, this results in a total of 630 hidden weights, 60 output weights and 16 bias values. To sum up, the model has 706 degrees of freedom. The two-layered RNN13-2L contains two hidden layers with 10 neurons and 3 recurrent layers. With 13 inputs, this results in a total of 830 hidden weights, 60 output weights and 26 bias values. Consequently, the size of the two-layered net is slightly larger than RNN13 with 5 recurrent layers per hidden layer. To sum up, the model has 916 degrees of freedom. Although LUT and RNN are quite different data-based models, the size seems comparable and within the same order of magnitude.

When running three models, i.e. LUT, RNN13 and RNN13-2L, with a frequency of 1000 Hz on a commercial Laptop (Lenovo T490s; Intel i7-8665 4 core processor; 16 GB DDR4 Ram), we obtain realtime factors between 7 and 30 percent. That is, we are at least three times faster than realtime. The tests result from the maneuvers of Figures 5.25 to 5.25 have a length between 6 and 12 seconds. In practice, intertwining the soil prediction models with the excavator model and deploying it on the realtime computer SCALEXIO, we experience realtime capable behavior simulating with 500 Hz.

Summary of the Discussion of the Numerical Examples We present three exemplary excavation cycles, consisting of a start phase a constant velocity phase and a lifting phase, compare Paragraph 5.4.1. The Lookup method fails at parts of these meneuvers. The RNNs cover the main aspects of the trajectories. In Figure 5.18 and the succeeding error plots, we observe, the absolute maximum error for longitudinal force FX, horizontal force FZ and torque TY. These contain all relevant aspects of the output data. While RNN13 has a higher absolute maximum error in the component FX, it has a lower error for FZ and TY, compared to RNN7. The LUT error is always higher as we expect it to be. The second study revealed that a two layered RNN with only three recurrent layers per hidden layer yield slightly better results. Especially for maxima and minima of forces FZ the approximation of the two hidden layer RNN yields better approximations, see Figure 5.27.

6 Discussion

We successfully implemented two classes of realtime capable, data-based algorithms and deployed them at a driving simulator. In this thesis, we analyzed the prediction quality based on test excavation cycles, recorded with operator signals from the excavator cabin. We successfully tested the online force prediction at the driving simulator. Operators feel a resistance force when the bucket enters the soil. This lays the foundation for more realistic simulator studies. Applications include operator training, and quantitatively accurate load predictions, enhancing the physics of the simulation.

6.1 Results of this Thesis

Within this thesis we studied three different classes of particle simulation algorithms, namely DEM on an acceleration level, NSCD on a velocity level and PBD on a position level. We presented a comparison of three representative algorithms of these classes and analyzed them with respect to efficiency, robustness, and applicability. The motivation behind this comparison was twofold. First, the application of particle based soil simulation in realtime with one of the particle simulation classes. PBD is a possible candidate, but fails in the accurate prediction of draft forces. Second, the application of one of these algorithms for offline simulations to train a data-based model. Penalty-based DEM, as implemented in GRAPE, is suitable for this purpose.

We covered a well-known approach for static forces, namely FEEs, and discussed extensions to a dynamic setup. In practice, the FEE extension to a dynamic setup was instable and not promising. We mentioned different alternative physical hybrid models which increase the performance of particle based methods. We decided to focus on data-based prediction algorithms and set up a DEM LUT approach. We examined different variants including TPWL and stochastic processes and implemented them prototypically for online prediction. The latter variant enables the reproduction of similar oscillation amplitudes and bares similarities to a SDE. We pointed out the connection and similarity to Ornstein-Uhlenbeck

processes. Additionally, we considered Neural Networks in the context of supervised learning. We examined and compared FNNs and the advantages of RNNs and formulated a second data-based algorithm. Both networks possess a Universal Approximation Property, so that in theory, given enough meaningful training data, we achieve arbitrary approximation quality. We successfully implemented the DEM LUT approach and the DEM RNN approach at an in-house driving simulator.

6.2 Outlook and Future Work

The assessment of draft forces in realtime, brings a variety of benefits and possible applications. The development of autonomous excavators with optimized trajectories can be studied. We can test new features within excavation prototypes in dangerous situations with differently skilled operators and replay and reproduce these tests. In a field experiment, reproducibility is difficult as external conditions change and the setup involving heavy machinery is expensive and cumbersome.

We plan to implement a terrain server, keeping track of the current terrain heights. We aim to visualize the digging process and decrease the terrain height accordingly.

We noticed that the reaction forces are at times to high for the hydraulic multibody model of the excavator. This causes undesired effects at the driving simulator and may cause the model to fail. We want to resolve this issue by either stiffening the hydraulic excavator model or by introducing filters to avoid high force peaks. A proper damping of the reaction forces might be necessary, also a limitation of the cutting depth. Furthermore, a convincing visualization is missing so far. Such a feature will enable a visual feedback of the digging process, with terrain height adaption and bucket filling. This will be a further step to enhance the operator experience during excavation. Prototypical studies have revealed that a terrain server with variable terrain height in Unity and the excavator model leads to promising results. We also aim to include the filling of the bucket in the visualization. We further want to incorporate the forces acting upon the bucket, when we have filled it and move it above the soil. One possibility is a variable mass placed in the center of gravity of the bucket, depending on the previous excavation cycle and the current trajectory.

We propose the extension o the DEM LUT approach with the vertical velocity $\boldsymbol{v}_z$ as fourth Lookup parameter. This might improve the approximation of the high vertical forces that arise when the soil cutting is not performed as it should be. The training data of the Recurrent Neural Network, although yielding good prediction quality, does not contain all necessary maneuvers. Collecting more trajectories and

excavation cycles of different operators and including further training data will increase model fidelity and accuracy.

The time-dependent nature of RNNs resembles the structure of the physical system. However, the recurrence also has an undesired effect. If the error in the current time step is high, this will influence also the succeeding time steps. A FNN has the advantage, that the error of the current time step has no influence on the output in the next time step. A thorough analysis on the propagation of errors in the RNN context is most welcome.

So far, we considered one specific soil, a mixture of sand and gravel. A study involving different material using the same excavation trajectories might result in a function

$$f \quad : \quad \mathbb{R}^6 \longrightarrow \mathbb{R}^6, \qquad (\boldsymbol{F}_A, \boldsymbol{T}_A)^T \longmapsto (\boldsymbol{F}_B, \boldsymbol{T}_B)^T,$$

mapping the computed forces and torques of the interaction with the first soil A to the second B. Furthermore, we think of a similar procedure for different excavation buckets. Thus, we avoid the expensive offline simulation for a new material and allow the force prediction for new soil by concatenation of one of the existing online models with the unknown function f.

Ultimately, a realtime capable reduced particle model, possibly running on GPUs would be most welcome.

A Appendix

A.1 Rotations in Multibody Dynamics

The following paragraph is based upon [Kle15b], [SE17] and [Woe16]. We first recall some general facts on quaternions before turning to unit quaternions for the description of rotations.

a) A quaternion $\boldsymbol{q} = (q_1, q_2, q_3, q_4)$ is a vector in $\mathbb{R}^4$. The constituents q_i are the scalar components of $\boldsymbol{q}$.

b) A quaternion can be expressed as

$$q = q_1 + q_2 i + q_3 j + q_4 k = q_1 + \boldsymbol{q^i}^T \boldsymbol{i}$$

with real scalar part q_1 and an imaginary part $\boldsymbol{q^i} = (q_2, q_3, q_4)^T$ with imaginary units $\boldsymbol{i} = (i, j, k)^T$. The imaginary units satisfy the relation $i^2 = j^2 = k^2 = ijk = -1$.

c) The sum of two quaternions $\boldsymbol{q} = (q_1, q_2, q_3, q_4)$ and $\boldsymbol{p} = (p_1, p_2, p_3, p_4)$ is defined componentwise as

$$\boldsymbol{q} + \boldsymbol{p} = (q_1, q_2, q_3, q_4) + (p_1, p_2, p_3, p_4) = (q_1 + p_1, q_2 + p_2, q_3 + p_3, q_4 + p_4).$$

d) The product of two quaternions $\boldsymbol{q} = (q_1, q_2, q_3, q_4)$ and $\boldsymbol{p} = (p_1, p_2, p_3, p_4)$ is defined as

$$\begin{aligned}
\boldsymbol{qp} =& (r_1, r_2, r_3, r_4) \quad \text{with} \\
r_1 =& q_1 q_1' - q_2 q_2' - q_3 q_3' - q_4 q_4' \\
r_2 =& q_2 p_1 + q_1 p_2 + q_3 p_4 - q_4 p_3 \\
r_3 =& q_1 p_3 + q_2 p_4 + q_3 p_1 - q_4 p_2 \\
r_4 =& q_1 p_4 + q_2 p_3 - q_3 p_2 + q_4 p_1
\end{aligned}$$

f) Quaternions form a 4-dimensional $\mathbb{R}$-algebra $\mathbb{H}$.

e) The conjugate of a quaternion $\boldsymbol{q} = q_1 + q_2 i + q_3 j + q_4 k$ is the quaternion $\boldsymbol{q}^\star = q_1 - q_2 i - q_3 j - q_4 k$. The norm is defined as $\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q}\boldsymbol{q}^\star} = \sqrt{\boldsymbol{q}^\star \boldsymbol{q}} = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}$

g) The inverse quaternion is given by $\boldsymbol{q}^{-1} = \frac{1}{\|\boldsymbol{q}\|}\boldsymbol{q}^\star$.

h) The derivative of a time dependent quaternion function $\boldsymbol{q}(t)$ can be written as follows

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \cdot (0, \boldsymbol{\omega})^T$$

i) Unit quaternions: For rotations in 3-dimensional euclidean space, unit quaternions

$$\mathbb{S} = \{\boldsymbol{q} \in \mathbb{H} \mid \quad \|\boldsymbol{q}\| = 1\}$$

are relevant. The constraint guarantees that we have 3 degrees of freedom, but unlike rotation matrices, quaternions have no singularities.

A.2 Universal Approximation of FNNs and RNNs

In the following, we will recapitulate the proof of Theorem 2.4.11.

Theorem (Universal Approximation of FNN)**.** Let f be a non-constant, continuous function $f : \mathbb{R} \mapsto \mathbb{R}^n$. Then $\Sigma^{I,n}(f)$ is uniformly dense on compacta in $C(\mathbb{R}^I; \mathbb{R}^n)$.

The proof is based on [HSW89] and [SZ06].

Proof. Let $K \subset \mathbb{R}^I$ be any compact set. For any $f : \mathbb{R} \to \mathbb{R}^n$ we consider functions

$$NN(x) = \sum_{j=1}^{J} V_j f(A_j(\boldsymbol{x})) \in \Sigma^{I,n}(f),$$

where $A_j : \mathbb{R}^I \to \mathbb{R}$ is an affine linear mapping and $V_j \in \mathbb{R}^{1\times n}$. The product between V_j and f is the Hadamard product.

0) $\Sigma^{I,n}(f)$ is an algebra on K.
Let therefore $NN_1, NN_2 \in \Sigma^{I,n}(f)$ be arbitrary and $\alpha \in \mathbb{R}$. Then

$$\begin{aligned}
\alpha NN_1 &= \alpha \sum_{j=1}^{J} V_j f(A_j(\boldsymbol{x})) \\
&= \sum_{j=1}^{J} \alpha V_j f(A_j(\boldsymbol{x})) \\
&= \sum_{j=1}^{J} \bar{V}_j f(A_j(\boldsymbol{x})) \in \Sigma^{I,n}(f).
\end{aligned}$$

That is, $\Sigma^{I,n}(f)$ is closed under scalar multiplication. Consider

$$NN_1 = \sum_{j=1}^{J} V_j f(A_j(\boldsymbol{x}))$$

and

$$NN_2 = \sum_{j=1}^{\tilde{J}} \tilde{V}_j f(\tilde{A}_j(\boldsymbol{x})),$$

and without loss of generality $J < \tilde{J}$.

$$\begin{aligned} NN_1 + NN_2 &= \sum_{j=1}^{J} V_j f(A_j(\boldsymbol{x})) + \sum_{k=1}^{\tilde{J}} \tilde{V}_k f(\tilde{A}_j(\boldsymbol{x})) \\ &= \sum_{i=1}^{J+\tilde{J}} W_i f(B_i(\boldsymbol{x})) \end{aligned}$$

Here, $W_i = \begin{cases} V_j & \text{if } i < J \\ \tilde{V}_{i-J} & \text{if } i > J \end{cases}$, $B_i = \begin{cases} A_j & \text{if } i < J \\ \tilde{A}_i & \text{if } i > J \end{cases}$.

Thus $NN_1 + NN_2 \in \Sigma^{I,n}(f)$.

1) $\Sigma^{I,n}(f)$ is separating on K.
If $\boldsymbol{x}, \boldsymbol{y} \in K, \boldsymbol{x} \neq \boldsymbol{y}$, then there exists an $A \in A^I$ such that $f(A(\boldsymbol{x})) \neq f(A(\boldsymbol{y}))$. Let $a, b \in \mathbb{R}, a \neq b$ and $f(a) \neq f(b)$. Pick $A(\cdot)$ with $A(\boldsymbol{x}) = a$ and $A(\boldsymbol{y}) = b$, that is set $\omega_1 = \frac{b-a}{y_1 - x_1}, \omega_2 = 0, \ldots, \omega_n = 0$, and $\theta = a - \omega_1 x_1$. Then

$$\begin{aligned} A(\boldsymbol{x}) = \langle \omega, \boldsymbol{x} \rangle - \theta = \frac{b-a}{y_1 - x_1} x_1 + a - \frac{b-a}{y_1 - x_1} x_1 = a \\ A(\boldsymbol{y}) = \langle \omega, \boldsymbol{y} \rangle - \theta = \frac{b-a}{y_1 - x_1} y_1 + a - \frac{b-a}{y_1 - x_1} x_1 = b - a + a = b \end{aligned}$$

Then $f(A(\boldsymbol{x})) \neq f(A(\boldsymbol{y}))$ and $\Sigma^I(f)$ is seperating on K.

2) $\Sigma^{I,n}(f)$ vanishes on no point on K.
There are functions $f(A(\cdot))$ that vanish at no point on K. Choose $b \in \mathbb{R}$ with $f(b) \neq 0$. Set $A(\boldsymbol{x}) = \langle \boldsymbol{0}, \boldsymbol{x} \rangle + b$ Then it holds that $f(A(\boldsymbol{x})) = f(b)$ for all $\boldsymbol{x}$ in the compact set K. This implies that for every $\boldsymbol{x} \in K$, the constructed $\Sigma(f)$ does not vanish at $\boldsymbol{x}$. Thus $\Sigma(f)$ vanishes at no point on K.

With Stone-Weierstrass, it follows that $\Sigma^{I,n}(f)$ is ρ_K-dense in $C(\mathbb{R}^I; \mathbb{R})$. □

Let us continue with Theorem 2.4.13.

Theorem (Universal Approximation of RNN)**.** Let us consider a dynamical system

$$\begin{aligned} \boldsymbol{s}_{t+1} &= g(\boldsymbol{s}_t, \boldsymbol{u}_t), \\ \boldsymbol{y}_t &= h(\boldsymbol{s}_t), \end{aligned}$$

where $g : \mathbb{R}^J \times \mathbb{R}^J \mapsto \mathbb{R}^J$ is Borel measurable and $h : \mathbb{R}^J \mapsto \mathbb{R}^n$ is continuous. Then there exists an element of $RNN(f)$, which approximates the dynamical system 2.13.

The proof is due to [SZ06].

Proof. 1) Approximate $\boldsymbol{s}_{t+1} = g(\boldsymbol{s}_t, \boldsymbol{u}_t)$ by $\boldsymbol{s}_{t+1} = f(A\boldsymbol{s}_t + B\boldsymbol{u}_t - \theta)$
Let $\varepsilon > 0$ and $f : \mathbb{R}^{\bar{J}} \to \mathbb{R}^{\bar{J}}$ a continuous sigmoid. Let further $K \subset \mathbb{R}^{\bar{J}} \times \mathbb{R}^{\bar{J}}$ compact and $\boldsymbol{s}_t, \bar{\boldsymbol{s}}_t, \boldsymbol{u}_t \in K$ for all time steps $t = 1, \ldots, T$. We use Theorem 2.4.11 so that for any measurable function $g : \mathbb{R}^J \times \mathbb{R}^I \to \mathbb{R}^J, (\boldsymbol{s}_t, \boldsymbol{u}_t)$ and for any $\delta > 0$, we find a feedforward neural net

$$\begin{aligned} NN(\boldsymbol{s}_t, \boldsymbol{u}_t) &= Vf(W\boldsymbol{s}_t + B\boldsymbol{u}_t - \bar{\theta}), \\ \sup_{\boldsymbol{s}_t, \boldsymbol{u}_t \in K} |g(\boldsymbol{s}_t, \boldsymbol{u}_t) - NN(\boldsymbol{s}_t, \boldsymbol{u}_t)| &< \delta \text{ for all } t = 1, \ldots, T. \end{aligned}$$

Here, $V \in \mathbb{R}^{J \times \bar{J}}, W \in \mathbb{R}^{J \times \bar{J}}$ and $B \in V \in \mathbb{R}^{\bar{J} \times I}$, describe weight matrices and $\bar{\theta} \in \mathbb{R}^{\bar{J}}$ describes a bias. As f is continuous and T is finite, there exists a $\varepsilon > 0$ such that $\bar{\boldsymbol{s}}_{t+1} = Vf(W\bar{s}_t + B\boldsymbol{u}_t - \bar{\theta})$, holds with $|\boldsymbol{s}_t - \bar{\boldsymbol{s}}_t| < \varepsilon$ for all t = 1, ..., T.

If we set $\boldsymbol{s}'_{t+1} = f(W\bar{\boldsymbol{s}}_t + B\boldsymbol{u}_t - \bar{\theta})$, this translates into $\bar{\boldsymbol{s}}_t = V\boldsymbol{s}'_t$ and with $A = WV$, we get

$$\boldsymbol{s}'_{t+1} = f(A\bar{\boldsymbol{s}}_t + B\boldsymbol{u}_t - \bar{\theta}). \tag{A.1}$$

2) Approximate $\boldsymbol{y}_t = h(\boldsymbol{s}_t)$ by $\boldsymbol{y}_t = C\boldsymbol{s}_t$

Let $\varepsilon > 0$, then there exists an $\delta > 0$ such that for $|s_t - \bar{\boldsymbol{s}}_t| < \delta$ it holds that $|h(\boldsymbol{s}_t) - h(\bar{\boldsymbol{s}}_t)| < \varepsilon$ by continuity of h. Hence, it is sufficient to approximate $\hat{\boldsymbol{y}}_t = h(\bar{\boldsymbol{s}}_t)$ by a linear function of the form $\bar{\boldsymbol{y}}_t = C\bar{\boldsymbol{s}}_t$ with arbitrary accuracy. The approximation theorem for feedforward neural networks, see Theorem 2.4.11, yields that we find $\bar{\boldsymbol{y}}_t = Nf(M\bar{\boldsymbol{s}}_t - \bar{\theta})$. Here $N \in \mathbb{R}^{n \times J}, M \in \mathbb{R}^{\hat{J} \times J}$ and $f : \mathbb{R}^{\hat{J}} \to \mathbb{R}^{\hat{J}}$ is a sigmoid and $\hat{\theta} \in \mathbb{R}^{\hat{J}}$. Similar to equation (A.1), we may write

$$\bar{\boldsymbol{s}}_t = V\boldsymbol{s}'_t \text{ and } \boldsymbol{s}'_{t+1} = f(A\boldsymbol{s}'_t + B\boldsymbol{u}_t - \bar{\theta})$$

We write

$$\begin{aligned}\bar{\boldsymbol{y}}_t =& Nf(M\bar{\boldsymbol{s}}_t - \hat{\theta})\\ =& Nf(MV\boldsymbol{s}'_t - \hat{\theta})\\ =& Nf(MVf(A\boldsymbol{s}'_{t-1} + B\boldsymbol{u}_{t-1} - \hat{\theta})).\end{aligned}$$

Thus, by applying once again Theorem 2.4.11, we obtain

$$\tilde{\boldsymbol{y}}_t = Df(E\boldsymbol{s}'_{t-1} + F\boldsymbol{u}_t - \bar{\bar{\theta}})$$

where $D \in \mathbb{R}^{n\times\bar{\bar{J}}}, E \in \mathbb{R}^{\bar{\bar{J}}\times\bar{J}}, F \in \mathbb{R}^{\bar{\bar{J}}\times I}$ and bias $\bar{\bar{\theta}} \in \mathbb{R}^{\bar{\bar{J}}}$. The function

$$f : \mathbb{R}^{\bar{\bar{J}}} \to \mathbb{R}^{\bar{\bar{J}}}$$

is continuous. Let us then write in matrix form with

$$\boldsymbol{r}_{t+1} = f(E\boldsymbol{s}'_t + F\boldsymbol{u}_t - \bar{\bar{\theta}}) \in \mathbb{R}^{\bar{\bar{J}}}.$$

Then we get

$$\begin{aligned}\begin{pmatrix}\boldsymbol{s}'_{t+1}\\ \boldsymbol{r}_{t+1}\end{pmatrix} =& f(\begin{pmatrix}A & 0\\ E & 0\end{pmatrix}\begin{pmatrix}\boldsymbol{s}'_t\\ \boldsymbol{r}_t\end{pmatrix} + \begin{pmatrix}B\\ F\end{pmatrix}\boldsymbol{u}_t - \begin{pmatrix}\bar{\theta}\\ \bar{\bar{\theta}}\end{pmatrix}),\\ \tilde{\boldsymbol{y}}_t =& \begin{pmatrix}\mathbf{0} & D\end{pmatrix}\begin{pmatrix}\boldsymbol{s}'_t\\ \boldsymbol{r}_{t\cdot}\end{pmatrix}\end{aligned}$$

Whence, we may summarize

$$\begin{aligned}\tilde{J} = \bar{J} + \bar{\bar{J}}, \quad \tilde{\boldsymbol{s}}_t = \begin{pmatrix}\boldsymbol{s}'_t\\ \boldsymbol{r}_t\end{pmatrix} \in \mathbb{R}^{\tilde{J}},\\ \tilde{A} = \begin{pmatrix}A & 0\\ E & 0\end{pmatrix} \in \mathbb{R}^{\tilde{J}\times\tilde{J}}, \quad \tilde{B} = \begin{pmatrix}B\\ F\end{pmatrix} \in \mathbb{R}^{\tilde{J}\times I},\\ \tilde{C} = \begin{pmatrix}0 & D\end{pmatrix} \in \mathbb{R}^{n\times\tilde{J}}, \quad \tilde{\theta} = \begin{pmatrix}\bar{\theta}\\ \bar{\bar{\theta}}\end{pmatrix} \in \mathbb{R}^{\tilde{J}}.\end{aligned}$$

Finally, we may write

$$\begin{aligned}\tilde{\boldsymbol{s}}_{t+1} = f(\tilde{A}\tilde{\boldsymbol{s}}_t + \tilde{B}\boldsymbol{u}_t - \tilde{\theta})\\ \tilde{\boldsymbol{y}}_t = \tilde{C}\tilde{\boldsymbol{s}}_t.\end{aligned}$$

□

B List of Figures

Bibliography

[Agg18] C. C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Cham, Switzerland: Springer Nature Switzerland AG, 2018 (cit. on p. 72).

[Alt16] H. W. Alt. *Linear Functional Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016 (cit. on p. 8).

[Arn73] V. I. Arnold. *Ordinary Differential Equations*. 10th edition. The Massachusetts Institute of Technology, 1973 (cit. on p. 11).

[Bal+16] M. Balzer, M. Burger, T. Däuwel, T. Ekevid, S. Steidel, and D. Weber. "Coupling DEM particles to MBS wheel loader via Co-Simulation". In: *Proceedings of the 4th Commercial Vehicle Technology Symposium, Kaiserslautern*. 2016, pp. 479–488 (cit. on p. 65).

[Bar12] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, Cambridge, 2012 (cit. on p. 66).

[Ben+16] N. Bennett, A. Walawalkar, M. Heck, and C. Schindler. "Integration of digging forces in a multi-body-system model of an excavator". In: *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics* 230.2 (2016), pp. 159–177. DOI: 10.1177/1464419315592081 (cit. on pp. 57, 86).

[Bur+17] M. Burger, K. Dressler, T. Ekevid, S. Steidel, and D. Weber. "Coupling a DEM material model to multibody construction equipment". In: *Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2017*. 2017, pp. 417–424 (cit. on p. 48).

[Bür20] J. Bürgi. *Progress Tabule*. 1620 (cit. on p. 15).

[Can99] H. N. Cannon. "Extended Earthmoving with an Autonomous Excavator". MA thesis. Carnegie Mellon University, 1999 (cit. on p. 57).

[Cou76] C. A. Coulomb. "Essai sur une application des règles de maximis & minimis à quelques problèmes de statique, relatifs à l'architecture". In: *De l'Imprimerie Royale, Paris* 7 (1776), pp. 343–382 (cit. on p. 57).

[CS79] P. A. Cundall and O. Strack. "A Discrete Numerical Model For Granular Assemblies". In: *Geotechnique* 29 (1979), pp. 47–65. DOI: 10.1680/geot.1979.29.1.47 (cit. on pp. 3, 30).

[Del34] B. Delauney. "Sur la sphère vide. A la memoire de Georges Voronoi". In: *Bulletin de l'Académiedes des Sciences de l'URSS. Classe des sciences mathématiques et na* 6 (1934), pp. 793–800. DOI: doi:10.1515/crll.1882.92.156 (cit. on p. 31).

[DM53] H. Deresiewicz and R. D. Mindlin. "Elastic Spheres in Contact Under Varying Oblique Forces". In: *Trans. ASME, Journal of Applied Mechanics* 20.3 (1953), pp. 327–344. DOI: 10.1115/1.4010702 (cit. on p. 41).

[DD04] A. Di Renzo and F. P. Di Maio. "Comparison of contact-force models for the simulation of collisions in DEM-based granular flow codes". In: *Chemical Engineering Science* 59.3 (2004), pp. 525–541. ISSN: 0009-2509. DOI: https://doi.org/10.1016/j.ces.2003.09.037 (cit. on pp. 29, 41, 42).

[dSPa] dSPACE. *dSPACE*. https://www.dspace.com/de/gmb/home.cfm. last accessed October 2021 (cit. on p. 94).

[dSPb] dSPACE. *SCALEXIO*. https://www.dspace.com/de/gmb/home/products/hw/simulator_hardware/scalexio.cfm. last accessed October 2021 (cit. on p. 94).

[Eva12] L. C. Evans. *An introduction to stochastic differential equations*. American Mathematical Society, 2012 (cit. on p. 12).

[Fen+09] Y. T. Feng, K. Han, D. R. J. Owen, and J. Loughran. "On upscaling of discrete element models: similarity principles". In: *Engineering Computations* 26.6 (2009), pp. 599–609 (cit. on p. 39).

[Fle09] F. Fleißner. "Parallel Object Oriented Simulation with Lagrangian Particle Methods". PhD thesis. Universität Stuttgart, 2009 (cit. on p. 42).

[For17] O. Forster. *Analysis 2*. 11th edition. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2017 (cit. on p. 11).

[Gho19] S. Ghorbani. "Simulation of soil-to-tool interaction using Discrete Element Method (DEM) and Multibody Dynamics (MBD) coupling". MA thesis. Iowa State University, 2019 (cit. on p. 41).

[Gib17] R. Gibaud. "Application of the Discrete Element Method to Finite Inelastic Strain in Multi-Material". PhD thesis. Institut polytechnique de Grenoble: Université Grenoble Alpes, Nov. 2017 (cit. on p. 52).

[GST12] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems*. Princeton, New Jersey: Princeton University Press, 2012 (cit. on p. 46).

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (cit. on p. 19).

[Gup15] P. Gupta. "Verification and Validation of a DEM-CFD Model and Multiscale Modelling of Cohesive Fluidization Regimes". PhD thesis. The University of Edinburgh: University pf Edinburgh, June 2015 (cit. on p. 38).

[Haa14] John K Haas. "A history of the unity game engine". In: (2014) (cit. on p. 33).

[HL15] E. Hairer and C. Lubich. *Numerical solution of ordinary differential equations*. Université of Genève and Universität Tübingen, 2015 (cit. on p. 12).

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. -Verlag Berlin Heidelberg, 1991 (cit. on p. 12).

[Her08] S. Herkt. "Model Reduction of Nonlinear Problems in Structural Mechanics: Towards a Finite Element Tyre Model for Multibody Simulation". doctoralthesis. Technische Universität Kaiserslautern, Dec. 2008 (cit. on p. 15).

[Her82] H. Hertz. "Über die Berührung fester elastischer Körper". In: *Journal für die reine und angewandte Mathematik* 92 (1882), pp. 156–171. DOI: `doi:10.1515/crll.1882.92.156` (cit. on p. 41).

[Hol14] D. Holz. "Parallel Particles (P2): A Parallel Position Based Approach for Fast and Stable Simulation of Granular Materials". In: *Workshop on Virtual Reality Interaction and Physical Simulation*. Ed. by Jan Bender, Christian Duriez, Fabrice Jaillet, and Gabriel Zachmann. The Eurographics Association, 2014. ISBN: 978-3-905674-71-2. DOI: `10.2312/vriphys.20141232` (cit. on p. 33).

[HAT15] D. Holz, A. Azimi, and M. Teichmann. "Advances in Physically-Based Modeling of Deformable Soil for Real-Time Operator Training Simulators". In: *2015 International Conference on Virtual Reality and Visualization (ICVRV)*. 2015, pp. 166–172. DOI: `10.1109/ICVRV.2015.64` (cit. on p. 3).

[HBK09] D. Holz, T. Beer, and T. Kuhlen. "Soil Deformation Models for Real-Time Simulation: A Hybrid Approach". In: *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2009)*. Ed. by H. Prautzsch, A. Schmitt, J. Bender, and M. Teschner. The Eurographics Association, 2009. ISBN: 978-3-905673-73-9. DOI: 10.2312/PE/vriphys/vriphys09/021-030 (cit. on p. 3).

[HSW89] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8 (cit. on pp. 17, 20, 117).

[HA90] U. Hunsche and H. Albrecht. "Results of true triaxial strength tests on rock salt". In: *Engineering Fracture Mechanics* 35.4 (1990), pp. 867–877. ISSN: 0013-7944. DOI: https://doi.org/10.1016/0013-7944(90)90171-C (cit. on p. 55).

[Jah] J. Jahnke. *DEM2D*. https://github.com/c2jahnke/DEM2D. last accessed October 2021. (cit. on p. 46).

[JSB19] J. Jahnke, S. Steidel, and M. Burger. "Soil modeling with a DEM Lookup approach". In: *PAMM* 19.1 (2019), e201900427. DOI: https://doi.org/10.1002/pamm.201900427 (cit. on pp. 4, 65).

[Jah+19] J. Jahnke, S. Steidel, M. Burger, S. Papamichael, A. Becker, and C. Vrettos. "Parameter identification for soil simulation based on the discrete elemente method and application to small scale shallow penetration tests". In: *PARTICLES 2019, VI International Conference on Particle-Based Methods. Fundamentals and Applications*. 2019, pp. 332–342 (cit. on pp. 4, 52, 81).

[Jah+20] J. Jahnke, S. Steidel, M. Burger, and B. Simeon. "Efficient Particle Simulation Using a Two-Phase DEM-Lookup Approach". In: *Multibody Dynamics 2019. IFToMM WC 2019. Computational Methods in Applied Sciences*. Ed. by A. Kecskeméthy. Vol. 53. 2020, pp. 425–432 (cit. on pp. 4, 65).

[JCB] JCB. *Prototype JCB 220X excavator powered by hydrogen fuel cell*. https://www.internationales-verkehrswesen.de/prototype-jcb-220x-excavator-powered-by-hydrogen-fuel-cell/. last accessed October 2021 (cit. on p. 2).

[KS05] I. Karatzas and S. E. Shreve. *Brownian Motion and Stochastic Calculus*. 2nd edition. Springer Science Business Media New York, 2005 (cit. on p. 70).

[Kle15a] M. W. Kleer. "Interaktive Fahr- und Betriebssimulation von Fahrzeugen, Land- und Baumaschinen - ein neuer Ansatz". PhD thesis. Fraunhofer ITWM, Kaiserslautern: Technische Universität Kaiserslautern, Oct. 2015 (cit. on pp. 4, 93).

[Kle] Michael Kleer. *RObot based Driving and Operation Simulator*. `https://www.itwm.fraunhofer.de/de/abteilungen/mf/technikum/rodos.html`. last accessed October 2021 (cit. on p. 2).

[Kle15b] J. Kleinert. "Simulating Granular Material using Nonsmooth Time-Stepping and a Matrix-Free Interior Point Method". PhD thesis. Stuttgart: Fraunhofer Verlag: Technische Universität Kaiserslautern, July 2015 (cit. on pp. 29, 30, 34, 35, 36, 37, 50, 81, 115).

[KOB13] J. Kleinert, M. Obermayr, and M. Balzer. "Modeling of Large Scale Granular Systems using the Discrete Element Method and the Non-Smooth Contact Dynamics Method: A Comparison". In: *Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2013*. 2013, pp. 191–200 (cit. on pp. 51, 66, 81).

[KP99] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. 3rd edition. Springer-Verlag Berlin Heidelberg, 1999 (cit. on p. 12).

[Kol07] D. Kolymbas. *Geotechnik*. 2nd edition. Springer-Verlag Berlin Heidelberg, 2007 (cit. on p. 52).

[Kre14] J. Krenciszek. "Proper Orthogonal Decomposition for Contact and Free Boundary Problems". PhD thesis. Dr. Hut-Verlag: Technische Universität Kaiserslautern, July 2014 (cit. on p. 15).

[KUK] KUKA. *KUKA*. `https://www.kuka.com/de-de`. last accessed October 2021 (cit. on p. 93).

[LeC+12] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: `10.1007/978-3-642-35289-8_3` (cit. on p. 17).

[LS80] D. T. Lee and B. J. Schachter. "Two algorithms for constructing a Delaunay triangulation". In: *International Journal of Computer & Information Sciences* 9 (1980), pp. 219–242. DOI: `https://doi.org/10.1007/BF00977785` (cit. on p. 31).

[LW92] H. Lutzdorf and M. Walter. *Jost Bürgi's Progress Tabulen (Logartihmen)*. 1992 (cit. on p. 15).

[Mac+19] M. Macklin et al. “Small Steps in Physics Simulation”. In: *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’19. Los Angeles, California: Association for Computing Machinery, 2019. ISBN: 9781450366779. DOI: 10.1145/3309486.3340247 (cit. on p. 33).

[Mat] MathWorks. *MATLAB Simulink*. https://de.mathworks.com/products/simulink.html. last accessed October 2021 (cit. on pp. 46, 68).

[McK85] E. McKyes. *Soil cutting and tillage*. Developments in agricultural engineering. Elsevier, 1985 (cit. on pp. 57, 59, 67).

[Mül+06] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. “Position Based Dynamics”. In: *Virtual Reality Interactions and Physical Simulations (VRIPhys) 2006*. Ed. by C. Mendoza and I. Navazo. 2006, pp. 1–10 (cit. on p. 32).

[Nat64] I. P. Natanson. *Theory of Functions of a real variable*. 3rd edition. New York: Frederick Ungar Publishing CO, 1964 (cit. on p. 10).

[OSu11] C. O’Sullivan. *Particlulate Discrete Element Modelling - A geomechanics perspective*. Spon Press, 2011 (cit. on pp. 30, 39, 41, 42).

[Obe13] M. Obermayr. “Prediction of Load Data for Construction Equipment using the Discrete Element Method”. PhD thesis. Aachen: Shaker Verlag: Universität Stuttgart, June 2013 (cit. on pp. 40, 44, 50, 54, 62, 77, 81).

[Obe+11] M. Obermayr, K. Dressler, C. Vrettos, and P. Eberhard. “Prediction of draft forces in cohesionless soil with the Discrete Element Method”. In: *Journal of Terramechanics* 48.5 (2011), pp. 347–358. ISSN: 0022-4898. DOI: https://doi.org/10.1016/j.jterra.2011.08.003 (cit. on pp. 42, 66, 67, 81).

[PPG11] B. P. Patel, J. M. Prajapati, and B. J. Gadhvi. “An excavation force calculations and applications: An analytical approach”. In: *International Journal of Engineering Science and Technology* 5.3 (2011), pp. 3831–3837. ISSN: 0975-5462 (cit. on p. 57).

[Ree64] A. R. Reece. “The Fundamental Equation of Earth-Moving Mechanics”. In: *Proceedings of the Institution of Mechanical Engineers*. Vol. 179. 1964, pp. 16–22 (cit. on pp. 57, 67).

[RAD05] M. Renouf, V. Acary, and G. Dumont. "3D frictional contact and impact multibody dynamics. A comparison of algorithms suitable for real-time applications". In: *Mutlibody Dynamics 2005, ECCOMAS Thematic Conference*. Ed. by J.M. Goicolea, J. Cuadrado, and J.C. Garcìa Orden. Madrid, Spain, 2005 (cit. on pp. 3, 37).

[Rew03] M. J. Rewieński. "A Trajectory Piecewise-Linear Approach to Model OrderReduction of Nonlinear Dynamical Systems". PhD thesis. Massachusetts Institute of Technology, June 2003 (cit. on p. 16).

[RHW86] D. E. Rumelhard, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. DOI: https://doi.org/10.1038/323533a0 (cit. on p. 17).

[SZ06] A. M. Schäfer and H. G. Zimmermann. "Recurrent Neural Networks Are Univsersal Approximators". In: *Artificial Neural Networks – ICANN 2006*. Ed. by Stefanos D. Kollias, Andreas Stafylopatis, Włodzisław Duch, and Erkki Oja. Vol. 179. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 632–640. DOI: 10.1007/11840817_66 (cit. on pp. 17, 19, 20, 117, 119).

[SE17] W. Schiehlen and P. Eberhard. *Technische Mechanik*. 5th edition. Wiesbaden: Springer Vieweg, 2017 (cit. on p. 115).

[SRS00] W. Schiehlen, A. Rükgauer, and T. Schirle. "Force Coupling Versus Differential Algebraic Description of Constrained Multibody Systems". In: *Multibody System Dynamics* 4 (4 2000), pp. 317–340. DOI: https://doi.org/10.1023/A:1009864502590 (cit. on p. 27).

[SA14] T. Schindler and V. Acary. "Timestepping schemes for nonsmooth dynamics based on discontinuous Galerkin methods: Definition and outlook". In: *Mathematics and Computers in Simulation* 95 (2014), pp. 180–199. DOI: 10.1016/j.matcom.2012.04.012 (cit. on pp. 46, 47).

[Sch99] A. Schinner. "Fast algorithms for the simulation of polygonal particles". In: *Granular Matter* 2 (1999), pp. 35–43. DOI: https://doi.org/10.1007/s100350050032 (cit. on p. 31).

[SSE10] R Seifried, W Schiehlen, and P Eberhard. "The role of the coefficient of restitution on impact problems in multi-body dynamics". In: *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics* 224.3 (2010), pp. 279–306. DOI: 10.1243/14644193JMBD239 (cit. on p. 42).

[SBN21] M. Servin, T. Berglund, and S. Nystedt. "A multiscale model of terrain dynamics for real-time earthmoving simulation". In: *Advanced Modeling and Simulation in Engineering Sciences* 8.11 (2021), pp. 1–35. DOI: https://doi.org/10.1186/s40323-021-00196-3 (cit. on pp. 3, 37).

[SW16] M. Servin and D. Wang. "Adaptive model reduction for nonsmooth discrete element simulation". In: *Computational Particle Mechanics* 3.1 (Mar. 2016), pp. 107–121. ISSN: 2196-4386. DOI: 10.1007/s40571-015-0100-5 (cit. on pp. 3, 60, 61, 62, 63).

[SS92] H. T. Siegelmann and E. D. Sontag. "On The Computational Power Of Neural Nets". In: *COLT* 10 (4 1992), pp. 440–449. DOI: 10.1145/130385.130432 (cit. on p. 20).

[Sin97] S. Singh. "The State of the Art in Automation of Earthmoving". In: *ASCE Journal of Aerospace Engineering* 10 (4 1997), pp. 179–208. DOI: 10.1061/(ASCE)0893-1321(1997)10:4(179) (cit. on p. 57).

[Soh03] H. Sohrab H. *Basic Real Analysis*. Springer Science+Business Media New York, 2003 (cit. on pp. 9, 18).

[Ste+21] S. Steidel, J. Jahnke, X. Chang, A. Becker, and C. Vrettos. "Triaxial compression and direct shear tests in the parametrization procedure of soil modeled with the Discrete Element Method". In: *PARTICLES 2021, VII International Conference on Particle-Based Methods. Fundamentals and Applications*. 2021, Accepted for publication (cit. on pp. 4, 52, 78).

[SM05] P. Stoica and R. Moses. *Spectral Analysis of Signals*. http://user.it.uu.se/~ps/SAS-new.pdf. Prentice Hall, 2005 (cit. on p. 82).

[TA11] A. Tasora and M. Anitescu. "A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics". In: *Computer Methods in Applied Mechanics and Engineering* 200 (Mar. 2011), pp. 439–453. DOI: 10.1016/j.cma.2010.06.030 (cit. on pp. 34, 35).

[WS21] E. Wallin and M. Servin. "Data-driven model order reduction for granular media". In: *Computational Particle Mechanics* (2021), pp. 1–14. DOI: https://doi.org/10.1007/s40571-020-00387-6 (cit. on p. 3).

[WD07] A. Wilkinson and A. DeGennaro. "Digging and pushing lunar regolith: Classical soil mechanics and the forces needed for excavation and traction". In: *Journal of Terramechanics* 44.2 (2007), pp. 133–152. ISSN: 0022-4898. DOI: https://doi.org/10.1016/j.jterra.2006.09.001 (cit. on p. 67).

[Woe16] C. Woernle. *Mehrkörpersysteme - Eine Einführung in die Kinematik und Dynamik von Systemen starrer Körper*. 2nd edition. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016 (cit. on pp. 23, 115).

Scientific Curriculum Vitae

Jonathan Jahnke, born in Freiburg im Breisgau, Germany

04/2018 - 03/2022	Doctoral candidate at Fraunhofer Institute for Industrial Mathematics ITWM, Division Mathematics for Vehicle Engineering and at the Department of Mathematics of the Technische Universität Kaiserslautern
04/2015 - 11/2017	Master's studies in Industrial Mathematics at the Department of Mathematics of the Technische Universität Kaiserslautern with specialization in Differential Equations and Scientific Computing and minor subject Physics Degree: Master of Science (M.Sc.)
02/2016 - 07/2016	Exchange semester at Dipartimento di Matematica "Federigo Enriques" of Università statale di Milano
10/2014 - 03/2015	Scientific Internship in the field of wind measurement at the engineering office Hrafnkel, Pressigny, France
10/2011 - 09/2014	Bachelor's studies at the Department of Mathematics at Ruprecht-Karls-Universität Heidelberg with focus on Algebraic number theory and minor subject Physics Degree: Bachelor of Science (B.Sc.)
06/2010	General higher education entrance qualification at the Theodor-Heuss-Gymnasium Freiburg

Wissenschaftlicher Werdegang

Jonathan Jahnke, geboren in Freiburg im Breisgau

04/2018 - 03/2022	Doktorand am Fraunhofer-Institut für Techno- und Wirtschaftsmatematik ITWM sowie am Fachbereich Mathematik der Technischen Universität Kaiserslautern
04/2015 - 11/2017	Masterstudium Technomathematik am Fachbereich Mathematik der Technischen Universität Kaiserslautern mit Schwerpunkt Differentialgleichungen und Anwendungsfach Physik Abschluss: Master of Science (M.Sc.)
02/2016 - 07/2016	Auslandssemester am Dipartimento di Matematica „Federigo Enriques“ der Università statale di Milano
10/2014 - 03/2015	Wissenschaftliches Praktikum im Bereich Windmessung im Ingenieurbüro Hrafnkel, Pressigny, Frankreich
10/2011 - 09/2014	Bachelorstudium Mathematik am Fachbereich Mathematik der Ruprecht-Karls-Universität Heidelberg mit Schwerpunkt Algebraischer Zahlentheorie und Anwendungsfach Physik Abschluss: Bachelor of Science (B.Sc.)
06/2010	Abitur am Theodor-Heuss-Gymnasium Freiburg

The simulation of soil-tool interaction forces using the Discrete Element Method (DEM) is widely established. In addition to an acceptable prediction quality, the efficient simulation of granular material on high performance clusters with modern parallelization strategies for the industrial applications is indispensable. Although, for relevant problem sizes such simulations are so far not real-time capable. Further on, the inclusion of the human-machine interaction at a driving simulator combined with soil-tool simulation poses many interesting research questions.

This thesis therefore strives for sufficient performance considering alternative models and algorithms to achieve real-time capability. First, different types of particle models are discussed regarding force accuracy and efficiency. The pros and cons are pointed out and the suitability for real-time applications is evaluated. Second, we present two machine learning algorithms which allow force predictions in real-time. The application at the in-house excavator simulator is discussed and the capability is shown using relevant numerical examples.

ISBN 978-3-8396-1835-6

Fraunhofer VERLAG

9 783839 618356