Coupling of Reinforcement Learning and DEM based digital Twins for Machine Control and Optimization

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Elektrotechnik und Informationstechnik
der Universität Duisburg-Essen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Fabian Westbrink
aus
Ahlen

Gutachter: Prof. Dr.-Ing. Steven X. Ding
Gutachter: Prof. Dr.-Ing. Andreas Schwung
Tag der mündlichen Prüfung: 11.10.2022

DOI: 10.17185/duepublico/77042
URN: urn:nbn:de:hbz:465-20230329-073457-4

Acknowledgments

First of all, I would like to sincerely thank Prof. Dr.-Ing. Andreas Schwung of the South Westphalia university of applied sciences for giving me the opportunity to take this special path. Over the last six years, I appreciated his continuous support, guidance, encouragement and hours of insightful discussions. His constructive suggestions to my research allowed me to give meaningful contributions to science and to complete this work satisfactorily. I would also like to thank Prof. Dr.-Ing. Steven X. Ding of the Institute for Automatic Control and Complex Systems (AKS) at the University of Duisburg-Essen, who always supported me with helpful advice and discussions and made it possible for me to complete this Ph.D.

Furthermore, I would like to thank my colleagues of the department of Automation Technology, starting with Gavneet Singh Chahda, Fernando Arévalo, Jan Niclas Reimann and all the others for the incredibly exciting and terrific time with you. I will always look back at this great time. Many thank to the AKS colleagues for an interesting and intensive exchange of knowledge, too.

At last, I am deeply indebted to my Pia who not only supported me during this demanding period but also revised this work with great effort.

Münster, October, 2022 *Fabian Westbrink*

Contents

Abstract

This dissertation deals with the coupling of reinforcement learning (RL) algorithms with digital twins based on the discrete element method (DEM). These digital twins are developed to act as environments to solve RL problems of machine control and parameter optimization. Due to the remarkable performance of modern RL algorithms and the versatility of DEM simulation, the coupling of these two fields opens up possibilities for solutions to many problems of modern machines or processes. In order to achieve a suitable coupling and handle the computationally slow DEM simulations, appropriate methodologies are developed. By applying these methodologies and state-of-the-art RL algorithms to two specific applications, the applicability of the entire approach is presented. In the first application, RL is used to solve the single- and multi-actuation task of the novel peristaltic sortation machine. Therefore, a DEM based digital twin is developed to properly represent the complex interaction of the individual parts of this machine. The second application deals with the problem of the DEM input parameter optimization which is always required to research new materials are researched with the DEM. A newly developed approach to optimize the parameters using RL leads to remarkable results and lower computation times. The developed approaches and methodologies of this dissertation are generally adaptable to other problems and contribute to the usage of the combination of RL and DEM in many other research fields.

List of Figures

List of Tables

List of Notations

Abbreviations

Abbreviation	Expansion
A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
ACER	Actor-Critic with Experience Replay
ACRL	Actor-Critic Reinforcement Learning
ANN	Artificial Neural Network
AoR	Angle of Repose
BP	Backpropagation
CEP	Courier Express Parcel
CFD	Computational Fluid Dynamics
COF	Coefficient of Friction
COR	Coefficient of Restitution
CORF	Coefficient of Rolling Friction
CPS	Cyber-Physical System
DDPG	Deep Deterministic Policy Pradient
DEM	Discrete Element Method

DNN	Deep Neuronal Network
DoE	Design of Experiments
DPG	Deterministic Policy Gradient
DQN	Deep Q-learning
ER	Experience Replay
FEM	Finite-Element Method
FIFO	First In First Out
FMI	Functional Mockup Interface
FRL	Feudal Reinforcement Learning
HIL	Hardware-In-the-Loop
HMI	Human Machine Interface
HRL	Hierarchical Reinforcement Learning
IIoT	Industrial Internet of Things
IPC	Industrial PC
LAMMPS	Large-Scale Atomic/Molecular Massively Parallel Simulator
LS	LIGGGHTS
MBD	Multi-body Dynamic
MDP	Markov Decision Process
MLP	Multilayer Perceptron
MORL	Multi-Objective Reinforcement Learning
MuJoCo	Multi-Joint dynamics with Contact
ODK	Open Development Kit
PDE	Partial Differential Equations
PLC	Programmable Logic Controller
pph	parcels per hour

PPO	Proximal Policy Optimization
PSM	Peristaltic Sortation Machine
RL	Reinforcement Learning
SARSA	State-Action-Reward-Sate-Action
SGD	Stochastic Gradient Decent
SPH	Smooth Particle Hydrodynamics
SQ-Function	Synthetic Q-Function
STL	Stereo Lithography
TD	Temporal Difference
TRPO	Trust Region Policy Pptimization

Mathematical notations

Notation	Description
$\forall$	For all
$\in$	Belongs to
$\subseteq$	A subset of
$\approx$	Approximately equal to
$\rightarrow$	Assignment
$\doteq$	Equality relationship that is true by definition
$\max\{\cdot\}$	Maximum of $\{\cdot\}$
$\min\{\cdot\}$	Minimum of $\{\cdot\}$
$\underset{x}{\operatorname{argmax}} f(x)$	A value of x at which $f(x)$ is maximal
$P\{\cdot\}$	Probability of $\{\cdot\}$
$\mathbb{R}^n$	Space of n-dimensional vectors
$\mathbb{R}^{n\times m}$	Space of n by m matrices
$\mathbb{E}[\cdot]$	Expection of $[\cdot]$

$\mathbb{E}_x[\cdot]$	Expection of $[\cdot]$ with respect to x
x	A scalar
$\mathbf{x}$	A vector
$\hat{\mathbf{x}}$	Estimate of vector $\mathbf{x}$
$\mathbf{X}$	A matrix
$\mathbb{R}$	A set of real numbers

1

Introduction

The increasing complexity and advancement of recent products and processes require special interdisciplinary developments in research and academics. One of them is the interdisciplinary combination of mechanical, electronic and information technologies which emerges to a smart mechatronic system, known as Cyber-Physical System (CPS) [1]. CPS is a combination of physical and computational resources which control and monitor the physical part of the system and affect each other. Considering the conventional control of the physical process with feedback-loops, the controller design must be based on appropriate models of the entire CPS with all its subsystems. The generation of these complex models for the model-based approach with deterministic formulas is often obtained by physical laws in Partial Differential Equations (PDE) [2]. Unfortunately, this generation can turn into a tedious task due to the nested and complex structure of CPS.

Therefore, advanced simulation-based modeling approaches and suitable simulation methods are developed. Simulations based on these methods virtually representing the physical counterpart of the CPS, provide an accurate and time-dependent description and are better known as digital twins [3]. Digital twins consist of simulation methods that rely on PDEs of the individual disciplines and can cover certain aspects of the CPS. But rather than solving the PDEs analytically, recent simulation methods solve the equations numerically. The modelling of CPS with these simulation methods results into a holistic simulation-based solution and can be integrated into advanced control schemes. To name some prominent examples, the Multi-Body Dynamics (MBD), the Finite Element Method (FEM), the Computational Fluid Dynamics (CFD), or the Discrete Element Method (DEM) are specialized for their specific subjects and are able to generate precise models of the desired system. Each of these methods has been developed to

optimally compute the models by defining a certain finite mesh or even meshless, whereby the degree of discretization decides about the accuracy of the model and the resulting computation time. In particular, the DEM as a meshless method has featured recently with good results in many different areas, is able to model large simulation domains and is therefore used and considered in this work to generate a simulation based solution of the desired problems [4].

The DEM was originally developed to simulate roundish particles scaling from atoms up to huge rocks. Instead of microscopically simulating the precise and tedious interaction and movement of each particle, the DEM has been developed to handle a large number of particles and their numerous interactions and collisions on a macroscopic level with sufficient accuracy [5]. The DEM is nowadays applied in various fields of the material handling or process industry to simulate, predict and optimise applications. But, instead of only using the DEM standalone, there is a trend to couple it with other methods in order to extend the scope of the DEM. For instance, a multi-way coupling of the DEM to the CFD or MBD allows to simulate the complex landing of a lunar lander, but also gives insights into the broad applications fields of the DEM [6].

By numerically computing high-dimensional PDEs, the DEM allows to simulate a large number of independent, but colliding bodies and is able to generate precise models of desired applications. However, the control of high-dimensional PDEs is constrained by standard control approaches and requires more advanced approaches [7]. Therefore, instead of simplifying the simulation model and reducing the dimensionality, which cause inaccuracies and additional efforts concerning following the later real-world implementation, this work defines a different way. The developed methodologies incorporate a simulation-based approach that is capable of dealing with highly complex simulations in their original state. In particular, the established methodologies are able to handle complex simulation methods and are applied to DEM simulations models which act as wholesome representations of certain applications modeled as digital twins. In specific, Reinforcement Learning (RL), as one leading-edge data-based approach, is incorporated to fully operate with the DEM-based digital twins.

RL, known as one branch of machine learning, is characterized by learning from consequences of self-made decisions rather than from recorded data. In RL a so-called agent observes and interacts with a dynamic environment and is trained to update its policy to make decisions that yield the highest rewards. Therefore, it suits perfectly for for problems where reliable training information are difficult, expensive or impossible to obtain [8]. Approaches and applications of RL have been investigated for decades and scientific research in this field accelerated even more by using deep neural networks as function approximators and corresponding computational power which results in acceptable training times. Following this trend, recent RL approaches have shown that they are able to achieve excellent training results in many application areas, for example successfully beating human experts in board games or achieved unprecedented

high-score playing advanced video games [8]. Current developments yield to achieve even more effective algorithms and are being rolled out to a myriad of applications to further establish RL [9].

By combining the DEM with its enormous possibilities and RL with its superior performances, this work creates a novel coupling that can be used for many types of applications for which there has been no acceptable solution until now. The applications of DEM models acting as digital twins may differ from large bulk good to complex parcel handling situations, in which RL supports the optimization of standard routines or allows to solve high-dimensional control tasks. However, the coupling of DEM and RL faces two major issues which need to be solved to allow a successful training of the RL approaches. First, the DEM simulations act as complex dynamic environments in the RL context. These environments consist of very detailed and high-dimensional models, are therefore difficult to train, require advanced learning algorithms and a vast number of training episodes. Second, the inherent slow DEM solver yields significant computation times for single time-steps. These computation times scale considerably concerning the training of RL algorithms with thousands of episodes, which makes holistic training with DEM simulation models impracticable.

Therefore, in this work several novel approaches and methodologies are developed which allow the optimization of digital twins consisting of DEM models with RL. In order to generally allow the use of DEM simulation models as dynamic environments that are integrated into the RL structure, a suitable DEM-RL coupling framework is developed. Furthermore, five methodologies are developed to feasibly train RL algorithms with very complex and slow DEM simulations. In combination these methodologies allow the training of RL algorithms with complex DEM based digital twins in reasonable computation times and thus yield to sophisticated results. The developed methodologies are also adaptable to other techniques and allow the incorporation of other simulation-based modelling approaches.

In order to verify the developed methodologies, they are carried out on two different applications, namely the control of the developed parcel transportation machine and the DEM parameter optimization problem. These applications not only show the applicability of the developed approaches, but also the usefulness and superiority of the DEM to generate digital twins of real applications and RL to solve sophisticated problems.

As a demonstrator of using the bionic principle of peristalsis and establish a novel way to gently transport and singulate parcels in bulk constellations, the Peristaltic Sortation Machine (PSM) has been developed. This machinery is a by-product of the presented work and was developed from scratch. The PSM has been fundamentally designed and is now acting as a functioning prototype. During the development and prototyping, the DEM was found to be an ideal tool to simulate the intricate behavior of this machine interacting with parcels. Therefore, the DEM is enhanced to simulate not only bulk behavior, but also the complex interaction of this

new type of machinery for parcel transportation. To correctly process parcels with this machine, a control system with the described methodologies is developed that takes the inherent special features and complex interactions of the peristaltic movements into account. Since the real machine is controlled by a Programmable Logic Controller (PLC), a RL-PLC implementation is developed additionally and provides the ability to deploy and train the developed RL frameworks on the real PSM.

As already mentioned, the DEM is primarily known in the fields of molecular to heavy bulk simulations and offers the possibility to simulate a broad range of different materials and processes. Nevertheless, the simulation of individual goods always requires proper calibration of the DEM input parameters to realistically simulate the real material. Since there is no universal quasi-standard for the DEM calibration and especially no optimization algorithm established, this works shifts the DEM optimization problem into the world of machine learning. More specifically, a reinforcement learning is applied to this second application and used to optimize DEM material parameters. A suitable multi-objective RL structure with a novel pre-training strategy is developed and trained to find optimal DEM input parameters which lead to a comparably fast calibration process. This is also the first attempt of using RL in the field of DEM parameter optimization. To reduce the number of required DEM simulations within the optimization process and thus save time during the entire calibration and optimization procedure, the developed pre-training strategy is applied to this problem.

The methodologies developed to manage the machine control of the highly complex peristaltic sorting machine as well as to significantly increase the performance of modern DEM optimization are fundamentally described. Applying the developed methodologies to the outlined applications shows fast training times and exceptional results. Apart from that, the developed methodologies are adaptable to other problems, especially in the field of simulation coupling, handling slow DEM simulations or generally applying reinforcement learning on various challenges.

1.1 Objectives of the Dissertation

The main objective of this work is the development of managing highly complex DEM simulation models by utilizing reinforcement learning methodologies. Virtual models of real applications are created using the DEM simulation and allow the use of corresponding digital twins as environments in the RL context. In order to achieve proper results and tackling the high-dimensional DEM environments a large number of training episodes are required intrinsically. In addition, the inherently computationally slow DEM solver results in an unfeasible training situation. Therefore, in this work, multiple methodologies are developed to control complex DEM simulations coupled as digital twins by reducing the complexity of the environments and speeding up the entire training with appropriate strategies.

More specifically the objectives of this work are stated as:

- Incorporation of DEM simulation-based digital twins in a suitable optimization framework. Handling of highly complex simulation models and the slow computational DEM solvers.
- Coupling of DEM simulations with RL. Generation of event-based interactions of DEM simulation environments through high-level language programming. Enabling RL algorithms to interact with dynamic DEM environments.
- Development of suitable methodologies to speed up the training of RL agents with parallel computing and pre-training strategies.
- Elaboration of methodologies to reduce the complexity of high-dimensional control tasks while adapting curriculum and hierarchical learning strategies.
- Simulation-based modeling of the demonstrator application mimicking the real PSM. Modeling of individual components with various novel DEM approaches. Development of suitable environments which properly reflect the dynamics and behavior of the machine.
- Development of a RL approach for optimizing the parameters for the problem of the DEM material parameter calibration. Development of an automated mobile calibration unit and determination of standard DEM calibration tests. Modeling of DEM environments for solving the optimization problem using modern RL approaches.

1.2 Outline of the Dissertation

The entire dissertation consists of six chapters. The individual content, objectives and contributions of each chapter are briefly described in the following.

Chapter 1: Introduction: This chapter introduces and describes the motivation for this work. Furthermore, the main objectives, desired contributions and the organization of this work are defined.

Chapter 2: State of the Art: In this chapter, the relevant works are reviewed and divided into the subjects of RL and DEM optimization. Since RL is becoming a vast and popular field, only the most influencing literature and works are described, which review the recent RL algorithms, benchmarks and industrial applications. In the subject of the DEM optimization, the general need for DEM calibration, suitable physical calibration tests and the current optimization strategies are discussed.

Chapter 3: Theoretical Basics: This chapter forms the fundamental basis of the rest of the work. It is divided into the sections of the feedforward neural networks, the broad sections of RL and the DEM as well as the basics of digital twins. Together with the previous chapter, all

basic works in the different areas are reviewed and consider the necessary literature and basic knowledge.

Chapter 4: Machine Control of the PSM: This chapter concentrates on defining and applying the developed methodologies to different tasks of the machine control of the PSM. Therefore, the development of the PSM and the machine characteristics are briefly explained. The digital twin of the PSM made by the DEM simulation is described while facing the mechanical parts, the parcel and transport film approximation and the entire composed digital twin. The developed novel methodologies of coupling the DEM, the iterative learning schedule, distributed actor-critic RL and hierarchical RL framework are fundamentally explained. Afterwards the training of RL agents with the developed methodologies to solve the single- and multi-actuation transportation tasks and the results of the training are shown. Finally, to allow the use of RL in real applications, the development of an RL-PLC implementation is described and discussed regarding a small example.

Chapter 5: DEM Parameter Optimization: This chapter is dedicated to the DEM parameter optimization as the second application for the coupling of RL and DEM. There, the different calibration procedures and the novel calibration unit are described. The developed optimization procedure is tested on three different materials using digital twins of real calibration tests. Therefore a novel multi-objective RL framework incorporating the developed pre-training strategy is applied to search for optimal material input parameters. Last but not least, the results also show the remarkable performance of the developed approach.

Chapter 6: Conclusion and Future Work: This chapter summarizes the entire work, specifies the major developments and contributions. Finally, possible future works that subsequently use the outcomes of this work are proposed.

2

State of the Art

This chapter reviews recent literature and methodologies dealing with the relevant topics of this work. Therefore the subjects are divided into the main categories of RL, DEM coupling and DEM optimization. The state-of-the-art of the subject of RL is defined of established RL algorithms, benchmarks and a few industrial applications. Recent works of coupling the DEM to other tools and software, as one integral part required in this work, is described in the second subject. In the subject of DEM optimization the state-of-the-art of physical calibration tests and optimization procedures are presented.

Some of the relevant literature are discussed later in the theoretical basics or in the individual sections in detail.

2.1 Subject: Reinforcement Learning

Artificial intelligence and machine learning recently got pushed using deep learning techniques and achieved prominent results in processing images, video, speech, etc. [10]. Reinforcement learning as one of the machine learning methodologies has also made impressive breakthroughs in various fields and moreover was able to beat professional human counterparts in many games [11]. Current research focus on developing effective algorithms, better performance in competing benchmarks and novel application fields, especially for industrial use cases.

The *deep*, in deep RL, firstly contributed by [12] with the developed deep Q-learning (DQN) algorithm is able to deal with continuous states and discrete actions and shows remarkable performances in many Atari games. Due to this development, many researchers focus on new algorithms such as deterministic policy gradient (DPG) using Monte Carlo methods by [13] or

deep deterministic policy gradient (DDPG) as one actor-critic algorithm by [14]. Both algorithms have opened up the possibility to handle continuous action spaces. Additionally, DDPG has shown remarkable performance with the then established Multi-Joint dynamics with Contact (MuJoCo) physics environment. In [15] the advantage actor-critic (A2C) and asynchronous actor-critic (A3C) algorithms with remarkable performance are presented. In the following the trust region policy optimization (TRPO) has been developed and is very effective for high-dimensional control tasks [16]. In comparison to TRPO, the proximal policy optimization (PPO) by [17] is much simpler to implement, has improved the data-efficiency and is one of the most promising RL algorithms for recent applications.

Results and performance benchmarks of RL algorithms and methods are often highlighted by applying them in prominent games. The works of [12], [18] are very popular in which agents were able to successfully play the popular Atari games. A RL agent playing the traditional but very complex game Go was published by [19] and defeated the world's best Go player. The so-called algorithm Alpha Go initially used supervised learning to gain fundamental game knowledge from amateurs players and learn advanced strategies from professional players. By purely self-playing, Alpha Go Zero as an enhancement of Alpha Go, speed up the training considerably and also achieved superior performance in playing Go [20]. Based on Alpha Go Zero AlphaZero was released which was able to defeat the world champions in the games of Chess, Shogi and Go [21]. To show the applicability of RL for real-time games AlphaStar was released by Deepmind and achieved superior performances by playing strategy games [22]. In response to this, OpenAI developed Five which is based on PPO to demonstrate its performance in the strategy game Dota 2 [23]. The Agent57, one of the latest developments of Deepmind, is able to remarkably perform not just in one game, but in all 57 Atari games by incorporating a batch of relevant methodologies in RL of the recent years [7].

Regardless of the remarkable performances in the described games, RL applications in industrial environments or sensitive areas e.g. interacting with heavy machinery or humans are still restrained. The usage of neural networks as function approximators always require proper tuning and the results are not always repeatable because of the random initialization of the networks. Even when assuming that the network is tuned properly two major hurdles of deep RL are postulated in [24]. First, the stochastic and *black-box* nature of the control policy causes uncertainties and makes it nearly impossible to understand why and how the agent has learned the control policy. Even in properly validated cases, the highly non-linear function approximation can still result in very different behaviors when the input conditions change slightly. Second, there exists no standardized RL software to directly deploy the agent for many industrial applications at the moment. Long and unfeasible training times or e.g. communication delays in the desired system cause problems of gaining sufficient performances [24]. In [25] the three bottlenecks of RL approaches are described, which are the sample efficiency, exploration &

exploitation and the generalization & reproducibility. Regardless of the desired application these bottlenecks need to be solved to achieve appropriate results and performances

However, researchers of various branches have discovered RL as a promising tool for explicit industrial applications and could thus overcome the aforementioned hurdles. Some industrial applications are robotic manipulation [26], incorporating vision systems [27], increasing of the energy efficiency with RL [28], RL based control of the temperature of heating, ventilation and air conditioning systems [29] or control of complex soft and flexible robotics [30]. Another successful example is the reduction of electrical energy of the Google data-centre by an average of 30% using recent machine learning and RL techniques [31]. Also in the field of advanced control engineering many contributions using types of RL have been made [32], [33]. To name some explicit examples of the control domain, the learning of state feedback controllers [34], optimal control of nonlinear systems [35], [36] or adaptive optimal control algorithms [37] have been investigated. In sum, RL techniques are applicable in various fields, but require suitable algorithms and environments which often consist of complex simulation models.

Especially the control of industrial robots with their manifold application areas is a prominent example for using RL. Besides very simple examples, the control of any type of robot is a tedious task and deals with influences caused by multiple contacts, frictions, or soft materials. These influences are difficult to model with first-order physics and therefore successfully tackled with RL, often in combination with conventional controller design [38]. When training with real-world environments, these influences are directly learned by the agent but tend to be problematic because of the slow data generation, resetting of environments and safety issues. Due to that, the training of RL agents has mainly focused only on simulation-based modeling. However, when applying the trained agents from the simulation to real-world environments, the *reality gap* between both environments produces inconvenient results [39]. Therefore, recent works focus on the sim-to-real transfer by reducing this gap by adding noise in the simulation with the domain randomization method, or train the agent with sources of the same feature space with the domain adaption method [40], [41].

However, sufficient handling of computational slow and complex environments is still an unsolved problem and leads to unfeasible long training durations. Therefore, this work is based on the state-of-the-art and develops methodologies to directly incorporate the computational slow and complex DEM simulation models. These methodologies ease the training of RL-DEM coupling by reducing the complexity of the RL task and also expedite the training significantly.

2.2 Subject: DEM Coupling

Usually, developed DEM simulations entirely run in the back-end of the used simulation tool. These encapsulated simulations are optimized to provide good results but are limited in sharing

these information. Commonly, plots and animations are used to evaluate and share the results. An automatic evaluation or an exchange of the interim results with appropriate tools is often not foreseen or hardly possible. Therefore the research is focused on the exchange of the DEM results with other simulation methods. DEM simulations have been coupled in co-simulation with e.g. the Multi-Body Dynamics (MBD) in [42]–[44], the (FEM) [45], [46] or often with the CFD in [47]–[49]. Another approach described in [50] defines a duality of particles and artificial neural networks and mimics equations of the DEM into neural network structures. In [51], a coupling of the DEM with industrial controllers such as PLC or Industrial PC (IPC) allows for virtual commissioning of e.g. bulk good processes.

In recent works especially the coupling of FEM-DEM has been further developed. This coupling closes the gap of analyzing structural mechanics with the FEM interacting with materials made of a large number of discrete elements such as soils or concrete [52], [53]. Since there is no commercial software available that consecutively gives the possibility to simulate both simulation methods together, corresponding couplings methods are developed to combine these different domains [54].

In [55] three different approaches are formalized which lead to a suitable coupling with the DEM, or with any other simulation methods. The first and simplest approach is the integrated coupling, in which inbuilt functions allow for corresponding coupling possibilities i.e. multiple simulation methods are integrated into one entire software tool. The second approach, mainly focusing on hardware optimized systems, such as special Hardware-In-the-Loop (HIL) devices, is achieved by network-based coupling. There, a data-exchange, e.g. via TCP/IP between multiple participants is established. The third approach is called program based, which means the local coupling of methods on the software level on the same machine. Program based coupling can be achieved by using simple exchange files, more sophisticated object-oriented coupling or by the standardized Functional Mockup Interface (FMI) [56], [57].

The described works show the state-of-the-art possibilities to couple the DEM with other simulation methods and tools. Nevertheless, to allow to use of the DEM in a data-based and machine learning context a suitable coupling framework is required and developed in this work.

2.3 Subject: DEM Optimization

The state-of-the-art of the DEM optimization is basically determined by developed calibration tests and suitable optimization strategies. Generally, the established tests are independent of the developed strategies and are also interchangeable. The basics of the DEM and the relevant literature are discussed in Section 3.4.

To ensure qualitative DEM simulations with conclusive results it is essential to sufficiently calibrate and optimize the DEM input parameters. Only calibrated parameters lead to consistent

results between the real applications and the DEM simulation models. But this parameter calibration and optimization requires expert knowledge and a significant computation time. Therefore, researchers developed a data-base for certain materials, which is unfortunately very limited and only suitable for a specific DEM tool [58]. Even though there are contributed values available for many materials, the approximation of the shape of the particles and the used contact models require a determination of the relevant material parameters. In [59] the internal coefficient of friction, the coefficient of restitution or the coefficient of rolling-friction are denoted as the relevant material parameters of the calibration.

Since the material parameters describe the behavior on the microscopic level, there are certain ways to directly measure them e.g. with Atomic Force Microscopy [60] or High-speed cameras [61]. But often these measurements are not feasible. As a result, macroscopic tests with meaningful measurements and an additional calibration procedure are used to define suitable parameters [62].

Typically, a DEM calibration starts with general analysis of the material to be examined. Important characteristics like the particle size, size distribution, the bulk density or friction values between the desired material and relevant surfaces are measured [63]. Furthermore, to find suitable DEM input parameters, a number of different laboratory tests are conducted, important key parameters are determined and the results are compared to surrogate DEM simulations. An optimization algorithm is then used to iteratively change the input parameters to find a suitable input parameter set which yields to matching results to the physically obtained results. However, there is no unique solution for the desired material. Instead, in [64], [65] the DEM calibration is denoted as an ambiguous problem, where multiple DEM input parameter sets yield to similar results.

In the calibration a variety of established physical tests are used to identify relevant material characteristics. Depending on the examined material, a batch of different tests is used. One important characteristic of powder and bulk material is the resulting Angle of Repose (AoR) [66]–[68]. This angle can be determined by performing experiments with e.g. a lifting cylinder, a rotating drum [69], a tilting box [70], a shear box [71] or a funnel tester [72]. A combination of some of the aforementioned tests is also known as draw-down test [64], which can identify multiple characteristics in a single test run. To measure the specific discharge behavior and the mass flow of the desired materials a hopper discharge [73] or auger dosing experiment [65] are used. Another methodology to analyze the material behavior is the shear test applied by the Jenike shear tester or the ring shear test [74]. The entire calibration procedure is either based on a single test or a variety of tests to identity different material characteristics.

Regardless of the applied physical tests, suitable surrogate DEM simulations and optimization procedure, models have to be developed to determine appropriate DEM input material parameters. The very basic optimization procedure is the trial-and-error method which almost always requires

a high number of iterations and long computation times. To find optimal input parameter sets with only a small number of iterations, sophisticated optimization algorithms are developed. These algorithms are e.g. based on the Design of Experiments (DoE) method in [75]–[77] or made with generic algorithms [78]. When combining DoE with the multivariate regression analysis a calibration of cohesion materials can be achieved [79]. But this optimization requires approximately 100 iterations to find suitable parameters, where the number of input-parameters decide about the number of necessary simulations. In [80] the developed generalized surrogate modeling-based calibration is used to calibrate materials within 21 simulation runs, but only considers two different material parameters. Another approach uses surrogate models for optimization purposes and is described in [81]. Optimization procedures based on machine learning techniques with scope of supervised learning and using artificial neural networks are presented in [81]–[84]. Another approach applying support vector machines to optimize the parameters is described in [85]. Approaches using Kriging also have been developed to further decrease the required number of iterations and therefore the entire computation time.

The state-of-the-art of the subject DEM optimization is defined by a variety of established DEM calibration tests and optimization strategies. However, DEM optimization is still a challenging task and requires a lot of computation time. Therefore, this work contributes the development of the automated mobile DEM calibration and optimization procedure which speeds up the process considerably and is based on the developed DEM-RL methodologies.

3

Theoretical Basics

In this chapter the fundamentals of the relevant subjects used in this work are described. The theoretical basics are divided into the sections of RL, with the relevant formulations and algorithms, a brief introduction into the topic of the digital twin and in the basics of the DEM explaining the complete simulation method. But first, the fundamentals of feedforward neural networks are explained which are a necessary standard tool in RL.

3.1 Feedforward Neural Networks

Artificial Neural Networks (ANN) consist of several layers i.e. input layer, hidden layer and output layer, where the information are propagated through all layers. If there is more than one hidden layer, the network is called Deep Neural Network (DNN). Within one layer there are one or multiple neurons which are connected to the surrounding layer with *weights*. In a fully connected feedforward network, also known as Multilayer Perceptron (MLP), built with a specific *width* (number of neurons per layer) and *depth* (number of hidden layers) as the very fundamental architecture, the information flow is only one-directional through the network [86]. DNNs can consist of multiple inputs $\mathbf{x}$ and one or multiple outputs $\mathbf{y}$, as shown in Figure 3.1. Feedforward networks are often used as function approximators, mapping inputs to corresponding outputs following the desired function [87]. But there are also other types of ANN architectures, for example convolutional neural networks and recurrent neural networks or many other sub-types which are used for various applications[88].

Each of the layers in the deep feedforward neural network is defined by the expression

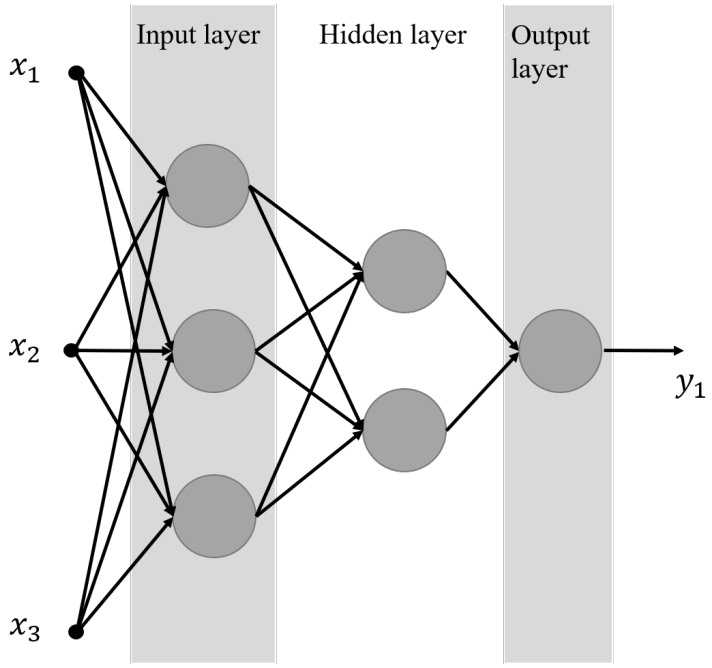

Figure 3.1: Structure of a feedforward DNN.

$$\mathbf{h}^{(i)} = \Theta^{(i)}(\mathbf{h}^{(i-1)}, \mathbf{W}^{(i)}), \tag{3.1}$$

for $i = 1, ..., m$ where m is the number of layers. $\mathbf{h}^{(i-1)}$ denotes the input and $\mathbf{h}^{(i)}$ the output of the current layer, $\mathbf{W}^{(i)}$ corresponds to the weight parameter and $\Theta^{(i)}$ is the chosen layer architecture or activation function. The activation function could be, for example, the Rectified Linear unit (ReLu), the hyperbolic tangent, the sigmoid or the the softmax function [86]. When considering m layers, the representation of each layer is given by

$$\begin{aligned} \mathbf{h}^{(1)} &= \Theta^{(1)}(\mathbf{h}^{(0)}, \mathbf{W}^{(1)}), \\ \mathbf{h}^{(2)} &= \Theta^{(2)}(\mathbf{h}^{(1)}, \mathbf{W}^{(2)}), \\ &\vdots \\ \mathbf{h}^{(m)} &= \Theta^{(m)}(\mathbf{h}^{(m-1)}, \mathbf{W}^{(m)}), \end{aligned} \tag{3.2}$$

with the weight matrices $\mathbf{W}^{(i)}$ which size depends on the number of neurons of the current and the following layer. The input $\mathbf{x} = \mathbf{h}^{(0)}$ is passed trough the network layers which approximates the desired output $\mathbf{y} = \mathbf{h}^{(m)}$. The learning of the network is accomplished by minimizing a certain loss function L for example determined by the squared error or the cross-entropy object function [86]. During the entire training, it is tried to minimize a specific cost function J, which is obtained by adding the losses and some kind of regularization term. Due to that, e.g. Backpropagation (BP) is used to compute the gradients of the loss function with respect to each

weight. The basic idea of BP is that the partial derivative of the loss with respect the weight parameters can be decomposed and thus the error can propagates back through the network [89]. The weights are then updated with gradient decent methods like Stochastic Gradient Decent (SGD) to minimize the loss [86]. In SGD not the true gradient as in Gradient Decent (GD), but instead a stochastic estimator of the gradient is used. Since SGD usually only uses minibatches of the entire training samples, it converges much faster compared to GD, but the minimizing of the error not as well as in GD [86].

3.2 Reinforcement Learning

Reinforcement Learning (RL) is the training of machine learning models and decision making while interacting with dynamic environments. Based on [90], RL can be also seen as the approach of computationally mimicking the intrinsic animal learning behavior which is based on trial-and-error learning. While interacting with an environment and taking actions, a learner/controller, also called the agent, learns to maximize the reward signal. For this process, not the immediate but rather the long term rewards are of interest. As shown in Figure 3.2 the agent receives an observation of the environment encoded in the state $s_t = s$ at each discrete time step $t = 0, 1, 2, ..., T$. Based on these information the agent decides for an action $a_t = a$ and as a consequence the environment updates at the time step $t + 1$ to the subsequent state s_{t+1}. At the same time the agent receives a reward $r_{t+1} = r \in \mathbb{R}$.

3.2.1 Markov Decision Process

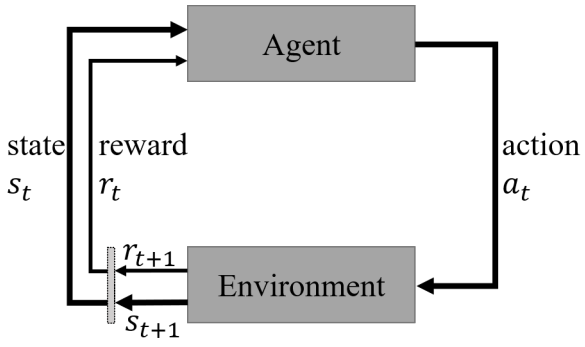

Figure 3.2: Reinforcement learning structure [90].

The basics of RL are formulated by considering an agent observing a specific environment which is modeled as a simple Markov Decision Process (MDP) [91]. The agent takes actions depending on its strategy, or also called policy, which maps observations to actions. The agent learns from these interactions while obtaining the rewards as shown in Figure 3.2. In the MDP the probability of reaching a new state and receiving a reward is given by a probability function

depending on the previous state and action. The overarching goal of the agent and the learning process is to maximize the cumulative reward.

In [90] the dynamics of the MDP are generally formalized as a 5-tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with:

- the set of all non-terminal states $s \in \mathcal{S}$,
- the set of all actions $a \in \mathcal{A}$,
- $P(s_{t+1}|s_t = s, a_t = a)$ as the probability transitioning into state s_{t+1} coming from state s and taking the action a at the time t,
- $r(s, a)$ as the immediate reward receiving advancing from state s to the following state s_{t+1} due to the action a,
- $\gamma \in (0, 1]$ as the discount factor presenting the different importance of future and immediate rewards.

In the learning process the agent observes the environment, often not fully observable, but only partially encoded in the state s. Based on this observation the agent takes an action a from the action set at each time step. The mapping of states to actions $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is defined as the policy of the agent. The stochastic policy

$$\pi_{\mathrm{s}}(a|s) = P(a_t = a|s_t = s), \tag{3.3}$$

is defined as the probability of taking an action a at a certain state s. The main objective of RL is to maximize the total discounted reward, also denoted as the return, valid for all timesteps $t < T$ and defined by

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{3.4}$$

with $0 \leq \gamma \leq 1$ as the discount factor [90]. This discount factor is usually set to be $\gamma < 1$ and leads to future rewards being less valuable compared to immediate results.

To achieve the maximum future discounted reward the optimal policy $\pi^*(s)$ has to be found. A policy π is defined to be better than another policy π' if the expected return is greater for all states and thus the optimal policy $\pi^*(s)$ is defined as

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi[G_t]. \tag{3.5}$$

To describe RL algorithms in detail, important definitions are to be explained. First, there is is a distinction between model-free and model-based learning methods. Generally, the methods

attempt to solve different problems, so that model-free methods rely on learning as their primary component while model-based methods rely on planning [90]. Model-based methods attempt to learn the transition probability and the reward function to plan trajectories without directly interacting with the environment. In model-free RL, for example temporal difference learning, the agent directly learns from experience while interacting with the environment but gains no knowledge about the environmental dynamics itself. In contrast, model-based methods construct a model of the environment to then learn indirectly within this constructed model.

Additionally, in RL two major approaches are distinguished, namely the value-based and policy-based RL which are explained in the following sections.

3.2.2 Value-Based Reinforcement Learning

RL using the value function for decision making has been established as a prominent approach and as a basis for modern RL techniques. The value function estimates how good it is for the agent to be in a given state and future rewards that can be expected. Therefore, value-based estimate the future rewards by evaluating the value function $V^{\pi}(s)$ as the expected return from the state s following the policy π. Formally V^{π}, also called the state-value function for the policy π is defined as

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|s_t = s], \tag{3.6}$$

$$= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right] \forall s \in \mathcal{S}, \tag{3.7}$$

in which $\mathbb{E}_{\pi}$ denotes the expectation over the episodes with the policy π [90]. Based on that, there exists an action-value function for a policy π, which defines the value of taking an action a in the state s under the policy π. This action-value function is similarly defined as

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a], \tag{3.8}$$

$$= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\right]. \tag{3.9}$$

To practically formalize the connection between the value of a state and future possible states, the return is recursively written as the immediate reward plus the discounted return. By that, as a form of dynamic programming, the value function results in

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|s_t = s], \tag{3.10}$$

$$= \mathbb{E}_{\pi}[R_{t+1} + G_t|s_t = s], \tag{3.11}$$

$$= \sum_{a} \pi(a|s) \sum_{s_{t+1}} \sum_{r} P(s_{t+1}, r|s, a) \left[r + \gamma \mathbb{E}_{\pi}[G_{t+1}|s_{t+1}]\right], \tag{3.12}$$

$$= \sum_{a} \pi(a|s) \sum_{s_{t+1}, r} P(s_{t+1}, r|s, a) \left[r + \gamma v^{\pi}(s_{t+1})\right] \forall s \in \mathcal{S}, \tag{3.13}$$

where the actions $a \in \mathcal{A}(s)$, the next states $s_{t+1} \in \mathcal{S}$, the rewards $r \in \mathcal{R}$. The probability $P(s_{t+1}, r|s, a)$, which is also known as the dynamics of the MDP, describes the one-step dynamics of the environment and is completely specified by this probability distribution for the next values of the state, reward given with the current state and action value. The equation 3.13 expresses a relationship between the value of a state and is known as the Bellman equation for V^{π} [90].

There exists at least one value function that maximizes the value of every state and is thus denoted as optimal value function

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \forall s \in \mathcal{S}, \tag{3.14}$$

and the optimal action-value function

$$Q^{*}(s, a) = \max_{\pi} Q^{\pi}(s, a) \forall s \in \mathcal{S}, a \in \mathcal{A}, \tag{3.15}$$

which gives the expected return for taking action a in the state s and thereafter following an optimal policy. Q^{*} can also be written in terms of V^{*} as follows

$$Q^{*}(s, a) = \mathbb{E}[r_{t+1} + \gamma V^{*}(s_{t+1})|s_t = s, a_t = a]. \tag{3.16}$$

The optimal value function in Equation (3.14) obeys a special policy and must satisfy the self-consistency condition. Additionally, as an important property it maximizes the expected value following from any action. Therefore, based on [90] the recursively Bellman optimality equation is expressed as

$$V^{*}(s) = \max_{a \in \mathcal{A}} \sum_{s_{t+1}, r} P(s_{t+1}, r_{t+1}|s, a)[r + \gamma V^{*}(s_{t+1})]. \tag{3.17}$$

The training algorithm of value-based RL follows the iterative cycle of policy evaluation and policy improvement. Within the policy evaluation the value function $V(s) \approx V^{\pi}(s)$ or $Q(s, a) \approx Q^{\pi}(s, a)$ is estimated for the current policy of the agent [92], which is then used to

improve the policy, following different strategies, for example greedily selecting actions with respect to the new value function. The improved policy is then evaluated and the cycle starts again. This procedure is the fundamental basis of value-based reinforcement learning [90].

The value function is accordingly updated with an appropriate backup operator. Model-free RL algorithms, such as Monte-Carlo or temporal-difference learning, update the value function by a sample-backup. There, the chosen action based on the current policy, the single state transition, and the reward are sampled from the environment which are then used to accordingly update the value function [92].

Following which strategy or choosing which action is another important definition of RL and is well-known as exploration-exploitation dilemma [90]. While training the agent, knowledge about the environment is accumulated and the agent has to decide between exploring the unknown parts of the environment (exploration) or pursuing the most promising strategy with the already gathered experience (exploitation). Suitable strategies carefully explore the environment and ensure that lesser-known parts of the environment are not as promising and then seek to increase the reward within the expected valuable parts.

Temporal Difference Learning

To allow an online policy update at the time step t, the so-called Temporal Difference (TD) method is explained. It uses the immediate reward to update the current policy without waiting for finishing the episodes. Hence, the TD method immediately updates the estimate of V with

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V s + 1 - V(s_t) \right], \tag{3.18}$$

on the transition from s to s_{t+1} and receiving r_{t+1} with the learning rate $\alpha > 0$. The target of TD methods can be seen as an approximation of the Bellman Equation in (3.17) incorporating the immediate and discounted reward. In [90] the pseudo-code of TD methods in Algorithm 1 is described.

Algorithm 1 TD(0) algorithm

Input: The Policy π to be evaluated
Data: Policy $\pi, \alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}$ arbitrarily, except $V(T) = 0$
loop for each episode:
  Initialize s_t
  loop for each step of episode:
    $a_t \leftarrow$ action given by π for s_t
    Take action a_t, observe r_{t+1} and s_{t+1}
    $V(s_t) \leftarrow V(s_t) + \alpha \underbrace{[\underbrace{r_{t+1} + \gamma V(s_{t+1})}_{\texttt{TD-target}} - V(s_t)]}_{\texttt{TD-error}}$
    $s_t \leftarrow s_{t+1}$
  end loop
  until s is not terminal
end loop

In each episode, for the current state s_t an action a_t is selected based on the policy π. The environment executes the action, gives the reward r_{t+1} and advances to the subsequent state s_{t+1}. The estimated value $V(s_t)$ is updated with the learning rate α and the TD-error, which is measured between the value at state s_t, the subsequent state s_{t+1} and any reward r_{t+1} accumulated along the way [92]. This TD method is called TD(0) or one-step TD because it estimates, contrary to n-step prediction, the following step only.

The classical TD method uses a look-up table consisting of the corresponding values and is thus computationally limited and only suitable for relatively small state spaces. In contrast to that, in deep RL, this table is replaced by function approximators e.g. feedforward neural networks estimating the value function.

SARSA algorithm

As a natural extension of the TD(0), the on-policy SARSA (State-Action-Reward-Sate-Action) using function approximators is described. On-policy learning algorithms are evaluated and improve the same policy which is used to gain the experience. In contrast to this, off-policy learning algorithms improve a policy that is different to the one which is used for choosing the actions.

The SARSA algorithm updates the value function according

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1} - Q(s_t, a_t)], \tag{3.19}$$

with the learning rate α for all non-terminal states s_t. The choice of action is often based on the ϵ-greedy-policy. ϵ-greedy is a combination of exploring by selecting random actions with the probability ϵ and exploitation selecting $\underset{a}{\operatorname{argmax}}\, Q(s,a)$ with the probability $1-\epsilon$ [90].

In the presented SARSA algorithm the action-value function $Q^\pi \approx \hat{Q}$ is represented by a parameterized function using a DNN. In RL the objective of using DNN is to approximate the value function $V^\pi(s) \approx \hat{V}(s;\mathbf{w})$ or state-action function $Q^\pi(s,a) \approx \hat{Q}(s,a;\mathbf{w})$ with a parameterized function with respect to the weights $\mathbf{w} \in \mathbb{R}^d$ which is a finite-dimensional weight vector. By adjusting the values of the parameter **w**, the target function is ideally approximated.

The SARSA algorithm is based on the general gradient-descent update for action-value predictions with the target U_t

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[U_t - \hat{Q}(s_t + a_t;\mathbf{w}_t], \tag{3.20}$$

where U_t can be any approximation of $Q^\pi(s_t,a_t)$, whereas for the semi-gradient SARSA algorithm it follows

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[r_{t+1} + \gamma\hat{Q}(s_{t+1},a_{t+1},\mathbf{w}_t) - \hat{Q}(s_t + a_t;\mathbf{w}_t]. \tag{3.21}$$

The pseudo-code of the entire one-step SARSA is shown in Algorithm 2 [90].

Algorithm 2 SARSA algorithm for Estimating $\hat{Q} \approx Q^*$

Input: Differentiable action-value function parameterization $\hat{Q}(s_t,a_t;\mathbf{w})$
Algorithm parameters: step size $\alpha > 0$, small $\epsilon > 0$
Initialize weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g. **w** = 0)

loop for each episode:
  $s_t, a_t \leftarrow$ initial state and action episode (e.g. ϵ-greddy)
  loop for each step of episode:
    Take action a_t, observe r_{t+1}, s_{t+1}
    if s_{t+1} is terminal **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha[r_{t+1} - \hat{Q}(s_t,a_t;\mathbf{w})]\nabla\hat{Q}(s_t,a_t;\mathbf{w})]$
      Go to next episode
    end if
    Choose a_{t+1} as a function of $\hat{Q}(s_t,a_t;\mathbf{w})$ (e.g., ϵ-greddy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha[r_{t+1} + \gamma\hat{Q}(s_{t+1},a_{t+1};\mathbf{w}) - \hat{Q}(s_t,a_t;\mathbf{w})]\nabla\hat{Q}(s_t,a_t;\mathbf{w})$
    $s_t \leftarrow s_{t+1}$
    $a_t \leftarrow a_{t+1}$
  end loop
end loop

3.2.3 Policy-Based Reinforcement Learning

In contrast to value-based methods, policy gradient methods directly optimize the policy $\pi(a|s,\boldsymbol{\theta})$ with gradient ascent to directly increase the reward per step. A value function can be learned as

well but is not required to determine the action. The policy $\pi(a|s, \boldsymbol{\theta}) = P(a_t = a|s_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta})$ describes the probability of taking an action a in the state s at the time step t with the parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ [90]. Additionally, these methods typically are of higher variance and therefore less efficient but they have a better convergence than value-based methods and can learn a parameterized policy even for continuous action spaces [92]. These methods seek to maximize the performance measure $J(\boldsymbol{\theta})$ with the objective function

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, J(\boldsymbol{\theta}), \tag{3.22}$$

and updating the parameter $\boldsymbol{\theta}$ with gradient ascent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta}_t), \tag{3.23}$$

where $\nabla J(\boldsymbol{\theta}_t)$ is the stochastic estimate of the gradient of the performance measure with respect to the parameter $\boldsymbol{\theta}$ [90].

The REINFORCE algorithm as one of the first policy-based RL algorithms is used to describe this approach [93]. Instead of using an entire episode, it uses a trajectory $\tau = s_t, a_t, s_{t+1}, a_{t+1}, ...$ to compute the performance measure

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{\tau} R(\tau)\pi_{\boldsymbol{\theta}}(\tau)\right], \tag{3.24}$$

with the cumulative reward over the trajectory assuming $\gamma = 1$

$$R(\tau) = \sum_{\tau} r_{t+1}, \tag{3.25}$$

and under the policy $\pi_{\boldsymbol{\theta}}$. By using the policy gradient theorem which is described in detail in [90], the derivative of $J(\boldsymbol{\theta})$ results in

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{\tau} R(\tau)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau)\right]. \tag{3.26}$$

The parameter $\boldsymbol{\theta}$ is then updated using gradient ascent according to Equation (3.23).

Methods that combine the approximation of policy and value functions are called actor-critic methods, where the actor is a reference to the learned policy while the critic estimates the value function [94].

3.2.4 Actor-Critic Reinforcement Learning

The actor-critic method combines the advantages of policy-gradient methods and the efficiency of value-based RL [92]. The Actor-Critic Reinforcement Learning (ACRL) architecture is shown in Figure 3.3 and consists of both actor and critic. The actor representing the policy determines an action a_t for the given state s_t. Additionally, the critic represents the value function and estimates the value $V(s)$ of the current state with the parameterized function of **w**, so that the actor defines the chosen action while the critic evaluates this action. Using the policy gradient theorem by [90] it follows

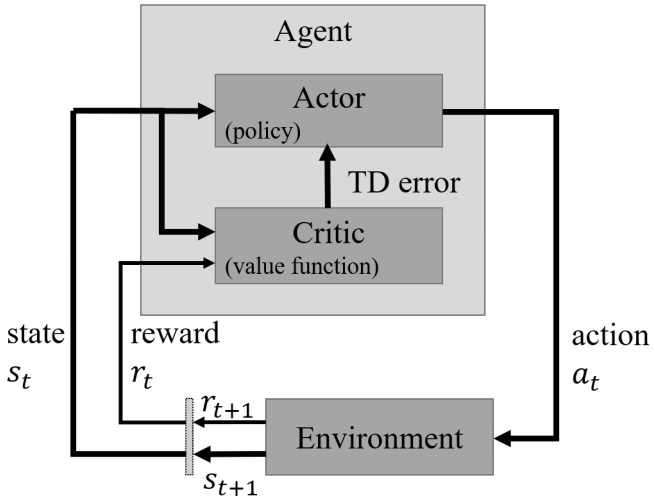

Figure 3.3: Actor-Critic architecture.

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{t=0}^{T} A^{\pi_{\boldsymbol{\theta}}}(s_t, a_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t), \tag{3.27}$$

with the advantage function $A^{\pi_{\boldsymbol{\theta}}}$ to be

$$A^{\pi_{\boldsymbol{\theta}}}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t), \tag{3.28}$$

which is approximated with the TD-error [94]

$$A^{\pi_{\boldsymbol{\theta}}}(s_t, a_t) = r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t). \tag{3.29}$$

Finally the policy parameter $\boldsymbol{\theta}$ is updated using gradient ascent. The actor selects an action a_t, advances in the subsequent state s_{t+1}, receives a reward r_{t+1} and updates the weight vector **w** according to Section 3.2.2.

Actor-critic methods incorporating the advantage function are called A2C and are suitable for discrete as well as continuous state and action spaces. A2C and the asynchronous version of it, A3C, which uses multiple independent agents interacting with copies of the environment in parallel, are stable and trained faster compared to other algorithms [15]. Based on the standard ACRL architecture several modern algorithms are developed which show improved performance in many RL tasks. One promising algorithm is the PPO which is characterized by ease of tuning and implementation, but comparably high-performance [17].

Proximal Policy Optimization

Generally, policy-gradient methods are prominent to sufficiently solve RL tasks using DNN for control or video games like Atari [12], but tend to be very sensitive to the chosen step-size. Small time steps dramatically increase the training time, whereas time steps could cause dramatic drops in the performance, if chosen to large. Thus, the strategy of constraining the size of the policy update is researched and adopted in algorithms like TRPO [16] or Actor-Critic with Experience Replay (ACER) [95]. But these methods are comparably complex and require intensive coding. Therefore, PPO is introduced which shows similar performance but with significantly lower implementation effort.

In PPO the policy update is limited in a small range using the clipped surrogate objective function. This objective function is an enhancement of the surrogate objective function used in TRPO which maximizes

$$L^{CPI}(\boldsymbol{\theta}) = \mathbb{E}[r_t(\boldsymbol{\theta})\hat{A}_t], \tag{3.30}$$

with $\boldsymbol{\theta}$ as the policy parameter,

$$r_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_{old}}(a_t|s_t),} \tag{3.31}$$

as the ration between the new and old policy and $\hat{A}_t$ as the estimated advantage at the time t [17]. The old policy $\pi_{\boldsymbol{\theta}_{old}}$ is the probability of taking an action a_t in the state s_t given the policy parameter $\boldsymbol{\theta}_{old}$ from the previous epoch and new policy π is therefore the probability given by the updated policy parameter $\boldsymbol{\theta}$. As a consequence of Equation (3.30), considering the new policy is much more probable compared to the old one, this leads to a large gradient step and policy update respectively. TRPO counteracts this massive update using the complex Kullback–Leibler (KL) divergence constraint [16]. In contrast to that, in PPO the surrogate object function simply clips the probability ratio of the surrogate model with

$$L^{CLIP}(\boldsymbol{\theta}) = \mathbb{E}[\min(r_t(\boldsymbol{\theta})\hat{A}_t, clip(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{3.32}$$

where ϵ is the hyperparameter of the clipping function. The minimum operator ensures that the updates are lower bound or pessimistic bound. Equation (3.32) handles two possibility ratios, a non-clipped and a clipped one, in the range of $1-\epsilon, 1+\epsilon$. If the advantage function is $\hat{A}_t > 0$, which refers to a comparatively good action choice, the probability will be increased to a maximum of $1+\epsilon$ to stabilize the training. If $\hat{A}_t < 0$, the ratio will be decreased consequently to a minimum of $1-\epsilon$ and the chosen action becomes less likely in the future. In general, PPO avoids updates which shift the policy too far away from the old policy.

The entire PPO algorithm as pseudo-code is given in Algorithm 3, using a fixed length of sampled mini-batches to accordingly update the policy [17]. In each iteration, each of the N actors collects data over the timesteps T. The NT data is sampled in mini-batches and is accordingly updated using SGD for K epochs [17]. As a result, PPO performs well in different benchmark tests, but compared to other algorithms requires with lower implementation effort and little use of hyperparameters.

Algorithm 3 PPO, Actor-Critic Style

for iteration=1,2,... **do**
  for actor=1,2,... **do**
    Run policy $\pi_{\boldsymbol{\theta}_{old}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1$,...,$\hat{A}_T$
  end for
  Optimize surrogate L wrt $\boldsymbol{\theta}$, with K epochs and mini-batch size $M \leq NT$
  $\boldsymbol{\theta}_{old} \leftarrow \boldsymbol{\theta}$
end for

3.2.5 Multi-Objective Reinforcement Learning

In Multi-Objective Reinforcement Learning (MORL) the optimization is not based on single but on multiple either conflicting, complementary, or independent objectives. The goal of MORL is to find one or more policies that optimize the multiple criteria simultaneously [96]. Enhancing the general form of the MDP, into a multi-objective optimization provides a vector of rewards instead of a scalar reward resulting in

$$\mathbf{r}(s,a) = (r_1(s,a)...r_{m_\mathrm{o}}(s,a)), \tag{3.33}$$

where m_o represents the number of objectives [96]. One approach for single policy MORL is the weighted sum or linear scalarization approach [97] generating a Synthetic Q-function (SQ-function) with

$$SQ(s,a) = \sum_{i=1}^{m_\mathrm{o}} \mathbf{w}_i Q_i(s,a), \tag{3.34}$$

where the weight vector satisfies the equation

$$\sum_{i=1}^{m_o} \mathbf{w}_i = 1. \tag{3.35}$$

Based on the SQ-function in Equation (3.34) the action which results in the maximal summed value is chosen. An adjustment of the individual weights sets the relevance for each objective. Additionally there exist a variety of different MORL approaches for single policies. As for example, [98] distinguishes between the W-learning, the analytic hierarchy process, the ranking and the geometric approach.

In the multi-policy MORL approaches tackle to find an optimal solution on the Pareto front incorporating trade-offs among multiple conflicting objectives [99]. The Pareto front represents the optimal set of a compromise solution which is dominating over other weak solutions. Establishing the true front of a multi-objective problem is rather impractical, thus the goal is to approximate. Therefore, a good approximation yields to solutions that are close to and evenly distributed along the front [99]. Hence, several RL algorithms employ an extended version of the Bellman equation and maintain the convex hull of the Pareto front as in [100], [101].

3.2.6 Hierarchical Reinforcement Learning

Solving highly complex problems, especially given the computational limits nowadays, requires suitable strategies. To solve a superior task, the Roman emperor Julius Caesar already stated *Divide et impera*, i.e. divide it into smaller sub-tasks and solve them successively. This strategy is inherently adapted in modern applications like positioning control of electrical drives with a cascade controller. This controller consists of different loops from the current control over speed control to the position control. This strategy is also applied to complex or long-horizon RL problems by dividing the task into levels of different ranks which are collectively referred to as the Hierarchical Reinforcement Learning (HRL). This approach has the potential to abstract multi-level control tasks and allows for transferability and interpretability of learned skills within the HRL structure [102].

As there are numerous of different HRL approaches developed, a formal definition generalizes and describes the different levels of the hierarchical structure [103]. The main complex task which is then divided into several levels and sub-tasks is denoted as Γ, its policy as π_Γ and one sub-task is denoted as ω. Thus, the policy of the sub-task is defined as π_ω and maps the states of the environment to the primitive actions or subjacent sub-tasks. Further, r_ω denotes the specific reward to train the policy of the sub-task π_ω, is individually defined and typically different from the reward associated with the main task. The reward is assigned to perform a certain sub-goal

g_ω which might be a state $s \in \mathcal{S}$ itself or an abstraction from a state [103]. Additionally, each sub-task has initiation conditions I_ω which may be a set of states in which ω can be chosen for execution and has certain termination conditions denoted as β_ω. The termination condition may be defined as the state to reach the sub-goal g_ω, as set of states in which ω should terminate if its being executed or a fixed time limit.

Considering any hierarchical structure, the sub-task of a certain level l is denoted as ω^l, which could also have an indicator if there are more than one sub-task per level. The policy of this sub-task is defined as

$$\pi^l_\omega : S \times \Omega^l \rightarrow [0, 1], \tag{3.36}$$

and chooses therefore a sub-tasks of the subsequent level [103]. Due to that, the action space referred to this sub-task is given by Ω^l, which is a set of sub-tasks of the lower level as for example $\omega^{l-1} \in \Omega^l$. If the sub-task is on the lowest level the set of sub-tasks is equal to the primitive action space as $\Omega^1 = A$.

Due to the different time episodes of the main task and the individual sub-tasks, HRL frameworks are usually based on the Semi-Markov Decision Processes (SMDP), which is an extension of the MDP, but involving the concept of different times for executions of sub-tasks or primitive actions [104]. Assuming starting from a state $s_t \in \mathcal{S}$, the agent chooses an action to achieve the sub-task $\omega^l_t \in \Omega^l$. The transition function of the SMDP is defined as

$$P(s_{t+c_{\omega^l_t}}, c_{\omega^l_t} | s_t, \omega^l_t) = P(s_{t+c_{\omega^l_t}} | s_t, \omega^l_t, c_{\omega^l_t}) P(c_{\omega^l_t} | s_t, \omega^l_t), \tag{3.37}$$

where $c_{\omega^l_t}$ denotes the number of timesteps ω^l_t is executed starting from s_t [102]. The reward obtained by performing a certain sub-task $R(s_t, \omega^l_t)$ is calculated with

$$R(s_t, \omega^l_t) = \mathbb{E}\left[\sum_{i=0}^{c_{\omega^l_t}} \gamma^i r(s_{t+1}, a_{t+1}) | s_t, a_t = \pi_{\omega^l_t}, (s_t)\right], \tag{3.38}$$

which indicates that the reward of a sub-task is equal to the expected cumulative reward accomplishing the sub-task while following the policy π^l_ω [103]. Finally, the optimal task policy would lead to the following desired Q-value with

$$Q(s_t, \omega^l_t) = R(s_t, \omega^l_t) + \sum_{s_{t+c_{\omega^l_t}}, c_{\omega^l_t}} \gamma^c_{\omega^l_t} P(s_{t+c_{\omega^l_t}}, c_{\omega^l_t} | s_t, \omega^l_t) \max_{\omega_t + c_{\omega^l_t}} Q(s_{t+c_{\omega^l_t}, c_{\omega^l_t}}, \omega^l_{t+c_{\omega^l_t}}), \tag{3.39}$$

$\forall s \in S$ and $\forall \omega^l \in \Omega^l$.

In HRL there exists a sub-task space as a super-set of all sub-tasks of the different levels which is summarized in

$$\Omega_{\mathrm{HRL}} = \{\Omega^2, \Omega^3, ..., \Omega^\Gamma\}. \tag{3.40}$$

To accomplish the main task, one or more agents must recursively train the policies to accomplish the subtasks through each layer. There is one policy π_{HRL} which maps the states from the highest level to the primitive actions following

$$a = \pi_{\mathrm{HRL}}(s), \tag{3.41}$$

and is also called the state-to-sub-task-to-action mapping [103]. The discounted cumulative reward of the entire HRL agent can be defined as

$$Q^{\mathrm{HRL}}(s_t, a_t) = \mathbb{E}_{a\sim\pi_{\mathrm{HRL}}|\Omega_{\mathrm{HRL}}}\left[\sum_{i=0}^{\infty}\gamma^{t+i}r(s_{t+1}, a_{t+1})|s_t, a_t\right], \tag{3.42}$$

where $a \sim \pi_{\mathrm{HRL}}|\Omega_{\mathrm{HRL}}$ indicates that a primitive action a is sampled following π_{HRL} to accomplish the available sub-task space Ω_{HRL} [103]. Hence, the optimal policy π^*_{HRL} and the optimal sub-task space Ω^*_{HRL} can be found with

$$\Omega^*_{\mathrm{HRL}}, \pi^*_{\mathrm{HRL}} = \underset{\Omega_{\mathrm{HRL}}}{\operatorname{argmax}}\ \underset{\pi_{\mathrm{HRL}}|\Omega_{\mathrm{HRL}}}{\operatorname{argmax}}\ Q^{\mathrm{HRL}}(s_t, a_t), \forall s \in S, a \in A. \tag{3.43}$$

Based on the previous definitions, several approaches are established which form the basis for recent developments. The main HRL approaches are the Options framework, [105], the feudal reinforcement learning (FRL) [106], the MAXQ decomposition [107] and the hierarchical abstract machines [108]. Here, only brief explanations of the options framework and FRL are given which are necessary to describe the developed HRL framework in this work. The principal idea of Options is the generalization of the concept of actions as a composition of sub-actions. In [105], Options is described with an agent interacting with a semi MDP, which is an extension of the general term of the MDP, but allows inconstant times between actions. Options consist three components, an intra-policy π_o, a termination condition $\beta : S \to [0, 1]$ and an initiation set $I \subseteq S$. This defined triple (I, π_o, β) is available in the state s_t if and only if $s_t \in I$ [105]. In options, the new states are observed and the initiation set is constantly checked. If the initiation set belongs to the current state, the option starts and the action is determined by the policy π_o instead of the global policy. As long as the termination condition β is false the action is determined by the global policy. Otherwise, the option stops, actions are determined

by the global policy and the initiation is checked for the next option. Therefore, the Options framework consists of a two-level hierarchy. In [109], the Options framework is enhanced with the policy-gradient theorems and thus the option-critic architecture is developed which is able to learn both, the internal policies and the termination conditions of options.

In FRL so-called super-managers lead managers and these managers, which again lead sub-managers etc. which represents the shape of a classical hierarchy pyramid. The managers of each level are allowed to set tasks, rewards or even give penalties to their sub-managers. In FRL two key principles form the basis of the feudal rules which are reward hiding and information hiding [106]. The rule of reward hiding defines that sub-managers have to obey their managers whether or not it satisfies the super-managers. Information hiding means that the superordinate level does not know the tasks which were defined by the lower levels for their subordinate, i.e. the super-manager does not know the of task of the manager for the sub-manager etc. The super-manager at the top-level decomposes the problem into smaller tasks which are then again split into even smaller sub-tasks. A novel approach facing FRL is developed by [110] and uses FeUdal Networks to automatically learn the sub-task while considering soft-conditions based on reward and information hiding.

3.3 Digital Twin

The digital twin, firstly introduced by NASA, is denoted as an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, etc., to mirror the behavior of its twin [111]. It is further known as a set of virtual information constructs that fully describes a potential or actual physical product of any type and size [112]. Creating a digital twin is an interdisciplinary challenge of generating surrogate models combining the physics, mechanics, electronics and information systems of the desired application.

Taking these surrogate models, [113] further distinguishes them into three categories namely the digital model, the digital shadow and the actual digital twin. The distinction is based on the assumption of the existence of a physical and its representing digital object. The digital model can contain a comprehensive description of the physical object but does not have an automated data exchange with the digital object. A possible state-change of the physical or digital object is thus not forwarded to the counterpart. In that regard the digital shadow in that regard exists of one-way data flow between the physical and digital objects in which the digital object follows the state of the physical one. Finally, the digital twin, based on a fully integrated data-exchange leads to a direct change of the state between the physical and digital objects and vice versa, where the two objects are mutually dependent.

Nowadays, the importance of a digital twin is crucial and especially required to reduce the

time-to-market and assist the development in all life cycles of the desired product. Today, a digital twin does not only consist of pure data, it rather is a combination of simulation models, structures, geometries, algorithms and machine learning which describe and simulate the real counterpart [113]. In the industry, the digital twins are usable for different kinds of research and development, life cycle estimation and manufacturing planning. While reflecting the status, performance, health or other specific characteristics of the physical counterpart, digital twins help to e.g. schedule preventive maintenance, understand how the system is performing, or observe the system dynamics for control purposes [114].

Digital twins support products over the complete life cycle. In each phase of the cycle, the digital twin can be used for different kinds of purposes from simulation-based modeling to remaining useful life estimation. [115] defines the digital twin as part of the digital world which is developed in parallel with the physical system or product and shown in Figure 3.4. Especially during the design and engineering phase the digital twin is an essential part and constantly developed and compared to the engineering process of the physical system. Even before a physical system is existent, the digital twin is used for pre- and feasibility studies. During the transition of the operation and service phase, digital twins become extremely important to gather and store product data, but it might happen that not all priorly developed models will be transferred into the new phase or have to be redesigned to solve the new tasks. However, special parts might be incorporated into the physical system for example or as an executable simulation as an assistant system [115].

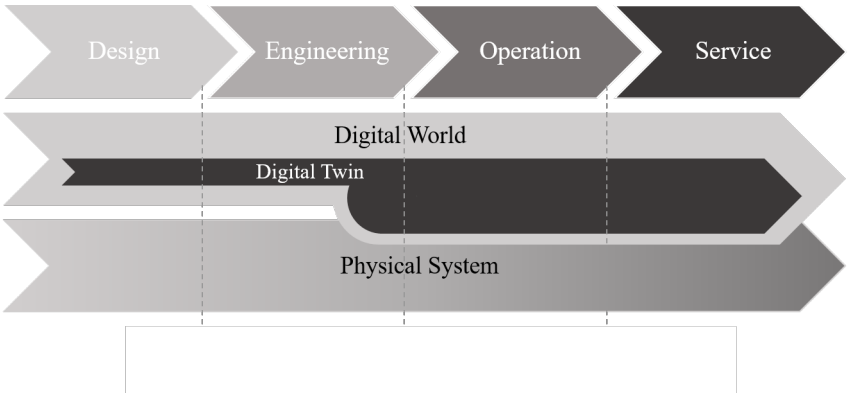

Figure 3.4: Digital Twin evolving along the phases of the product life cycle [115].

As an extension of the general term digital twin, i.e. as a one-to-one virtual replica, [116] proposes a combination of simulation technologies, model-based engineering and industry 4.0 to so-called experimental digital twins. These twins are supposed to close the gap between system engineering and complex simulations to ensure virtual test-beds, e.g. the development of suitable control algorithms [117].

To use digital twins in a machine learning or data-driven context, [118] formalizes the requirements and needs of standardized functions and interfaces. More specifically, to allow the

digital twin to be applicable as an environment in RL applications, a data exchange of states and rewards as well as interfaces for step or reset functions are necessary. Some promising works combining digital twins with RL are for instance learning of lifting weights with a humanoid robot and expedite the training through a virtual digital twin [119], deploying RL to inner loop flight control of an unmanned aerial vehicle [120] or [121] which create a digital twin for a RL based scheduler in a manufacturing environment.

3.4 Discrete Element Method

The simulation of surrogate models with multiple bodies, which are linked with constraints, independently movable, applied with forces, or in contact with static or dynamic structures is a challenging and complex task. Therefore, different simulation methods were developed, each of them specialized for a specific scope. On the one hand there are methods partially solving a problem in a predefined mesh as the Finite-Element Method (FEM), or the Computational Fluid Dynamics (CFD). On the other hand meshless methods like Smooth Particle Hydrodynamics (SPH) and the Discrete Element Method (DEM) [4] are used. Mesh-based methods face the problem of being dependent on the mesh size and structure and can therefore lead to differing results. As opposed to this, the DEM proves to be an ideal tool for creating a surrogate model with a huge number of independent but contacting bodies. But its weakness lies in its limited performance and its long computation times [122].

3.4.1 Basics of the Discrete Element Method

The Discrete Element Method (DEM) was originally developed by Cundall and Strack in the 1970s ,as a numerical method, capable of simulating particle behavior with an underlying contact model [123], [124]. Due to the historical background, the observed goods in the DEM are typically called particles, the term is therefore adopted in this dissertation. Regarding the computational power in the 1970s, the number of particles in the simulation domain was limited to a few thousands. Nowadays, due to increasing computational power, larger systems with millions of particles can be calculated. The DEM is commonly used to simulate the behavior of a broad range of materials. Simulations ranging from atoms and molecules [125], over typical bulk material like stones or coals [126]–[128], concrete fracture behavior determination [129], mechanical applications [55], [130], to the soil of the moon [131], [132], or even parcels and packages [133] are made by the DEM. Additionally, the DEM supports to analyze particular material behavior and therefore assists to optimize e.g. dosing of bulk good materials [134].

Inherently, the DEM provides valuable data about translational and rotatory particle movements as well as information about interactions with neighboring particles or walls. Likewise the particle mass-flow, the occurring velocities and forces, bridging, or the wearing of mechanical

parts can be determined [135], [136]. Additionally, the DEM can obtain trajectories, transient forces, or heat transfer of specific particles as well as atom charge. Information about damaging and a breaking behavior can be simulated in the DEM, too [137], [138].

DEM Cycle and Contact Model

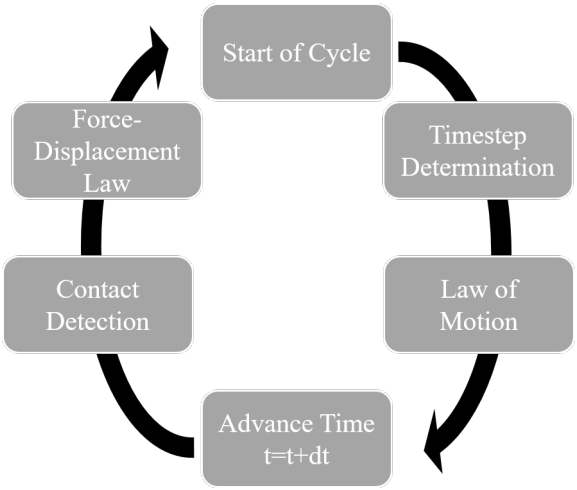

Figure 3.5: DEM computation cycle [139].

The fundamental methodology of the DEM follows the calculation cycle shown in Figure 3.5 and described in [139]. First the calculations require a valid, finite timestep which ensures the numerical stability of the model. The timestep is determined to properly calculate the behavior of contacting particles and is also used in the contact determination. The Rayleigh- and Hertz criteria, considering the material properties and presuming velocities, are used to obtain an optimal timestep [140]. In the next step, the positions and velocities of all particles are updated by Newton's laws of motion considering force and momentum calculations of the previous cycle. After that the timestep is accordingly advanced, and the current position of all particles in the simulation domain is checked by a contact detection algorithm creating a neighboring list consisting of close particles or obstacles. Possible contacts are calculated using certain contact models. Then forces and moments are updated regarding the force-displacement law [141]. This DEM cycle is repeated until the maximum timestep is reached.

The mathematical background of the DEM is the Lagrangian method. The trajectory of each particle inside the simulation domain is computed and updated accordingly. The force balance of each particle and the new position and orientation, described in [142] is given with

$$m_i \ddot{\mathbf{x}}_i = \mathbf{F}_{i,\mathrm{n}} + \mathbf{F}_{i,\mathrm{tg}} + \mathbf{F}_{i,\mathrm{f}} + \mathbf{F}_{i,b}, \tag{3.44}$$

$$I_i \dot{\boldsymbol{\omega}}_i = \mathbf{r}_{i,c} \times \mathbf{F}_{i,\mathrm{tg}} + \mathbf{T}_{i,r}. \tag{3.45}$$

The index i represents the corresponding particle with its mass m_i, tangential acceleration $\ddot{\mathbf{x}}_i$, inertia I_i, angular acceleration $\dot{\boldsymbol{\omega}}$ and the radius $\mathbf{r}_{i,\mathrm{c}}$ in the simulation domain. In Equation

(3.44) and (3.45), $\mathbf{F}_{i,\mathrm{n}}$ represents the normal force between particles, $\mathbf{F}_{i,\mathrm{tg}}$ is the force which acts tangential on the particles, $\mathbf{F}_{i,\mathrm{f}}$ describes the influence of surrounding fluids on the particles and $\mathbf{F}_{i,\mathrm{b}}$ summarises additional forces which act on the particles, e.g. gravity or magnetic forces. Eventually, torque which acts on the particles, is represented by $\mathbf{T}_{i,\mathrm{r}}$.

Geometrical shapes like spheres or well-defined volumes represent the mathematical particle. Since there is usually more than one particle in the simulation setup, contacts between particles could occur. Generally, the contact definition is distinguished and described considering either the hard- or soft-sphere model [143]. Unlike the contact of hard particles, soft particles allow slight overlap. The resulting force can be then derived from the overlap depth $\boldsymbol{\delta}_{\mathrm{n}}$ between the particles i, j and the velocity at the contact point $\Delta\boldsymbol{u}_{\mathrm{n}}$. Figure 3.6 shows a simple example of a

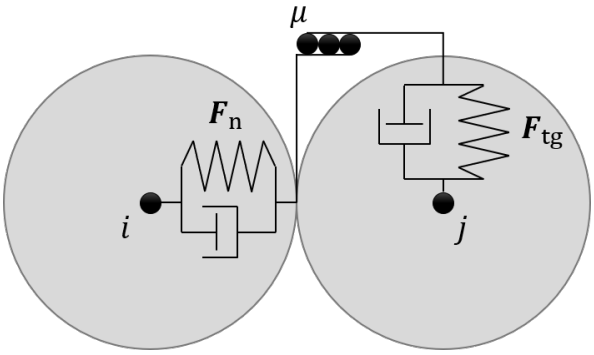

Figure 3.6: Simple spring-damper contact-model with friction.

spring- damper soft-sphere model with friction. Within this model the normal and tangential forces $\boldsymbol{F}_{\mathrm{n}}$ and $\boldsymbol{F}_{\mathrm{tg}}$ are given by

$$\boldsymbol{F}_{\mathrm{n}} = -k_{\mathrm{n}}\boldsymbol{\delta}_{\mathrm{n}} + c_{\mathrm{n}}\Delta\boldsymbol{u}_{\mathrm{n}}, \tag{3.46}$$

$$\boldsymbol{F}_{\mathrm{tg}} = \min\left\{\left|k_{\mathrm{tg}}\int_{t_{\mathrm{c},0}}^{t}\Delta\boldsymbol{u}_{\mathrm{tg}} + c_{\mathrm{tg}}\Delta\boldsymbol{u}_{\mathrm{tg}}\right|,\ \mu F_{\mathrm{n}}\right\}, \tag{3.47}$$

where $\Delta\boldsymbol{u}_{\mathrm{tg}}$ describes the relative tangential velocity the of contacting particles. The normal and tangential coefficients of spring and damper are represented by k_{n}, k_{tg}, c_{n} and c_{tg}. The Equations (3.46) and (3.47) are valid for particle-particle and particle-wall contacts. The coefficient of friction is represented by μ. Equation (3.47) consists of two parts. An integral part, which describes the elastic tangential motion stored in the spring. It allows to represent the deformation of the surface of the particle that occurs when the particles touch at $t = t_{\mathrm{c},0}$. The second part describes the damper system and thus the tangential energy dissipation. The Coulomb friction limits the tangential force. The exceedance of this limit leads the particles to slide over each other [142].

3.4.2 Particle Descriptions

From its origin of simulating atoms and molecules, the standard shapes of particles in the DEM are circles in the two dimensional and spheres in the three-dimensional space [126]. But the upcoming trend of using the DEM for various branches and technologies requires modeling besides simple roundish objects.

Shape Approximations

One simple solution is the usage of the rolling friction parameter to mimicry slightly non-circular objects. However, in nature, particles are mostly non-spherical and behave completely differently from spherical particles and thus are not possible to map with rolling friction only[144].

Therefore, there are several established approximation methods to allow the DEM to simulate complex shapes. Two of them are known by name as the multi-sphere approach and the approximation with superquadrics. The multi-sphere approach follows the idea of clumping several spheres to a conglomerate approximating a desired complex shape [145]. These clumped spheres, possibly varying in diameter and overlapping each other, are fixed at a distinct position in a local coordinate system. Sine the multi-sphere is an extension to the normal spheres in the DEM, the standard equations of the contact definition, Equations (3.46) and (3.47), and the contact detection of spherical objects in the simulation domain are retained. A multi-sphere

Figure 3.7: Multi-sphere approach with the original shape (left) and the approximation (right).

approximation is depicted in Figure 3.7 in which a stone, represented in an STL (Stereo-Lithography) format is approximated by just 37 clumped spheres. But approximating the desired shapes with a high resolution using the multi-sphere approach could end in a large number of required spheres per conglomerate. Subsequently, with an raising number of spheres, the computation time of the DEM increases and thus a realistic approximation with multi-spheres is often not feasible.

Finding an optimal number of spheres, fitting diameters and positions of single spheres within the conglomerate require appropriate methods and algorithms. One method which fills the desired shape with a random number of spheres and clusters them to a smaller number of resulting spheres is described in [146]. Another promising method to cluster to an optimal number of spheres with the off-volume criteria, a ratio between the calculated off-volume of the

resulting conglomerate and the original volume of the input shape, as well as fuzzy c-means clustering is introduced in [147].

The second method using superquadrics makes it possible to approximate shapes with only one particle. Combining concave and convex forms allows to model a high range of possible complex shapes. Superquadric particles are introduced and described in [144], [148], [149] with

$$f(x) = \left(\left|\frac{x}{a}\right|^{n_2} + \left|\frac{y}{b}\right|^{n_2}\right)^{\frac{n_1}{n_2}} + \left|\frac{z}{c}\right|^{n_2} - 1 = 0, \tag{3.48}$$

$$x = (x, y, z)^T,$$

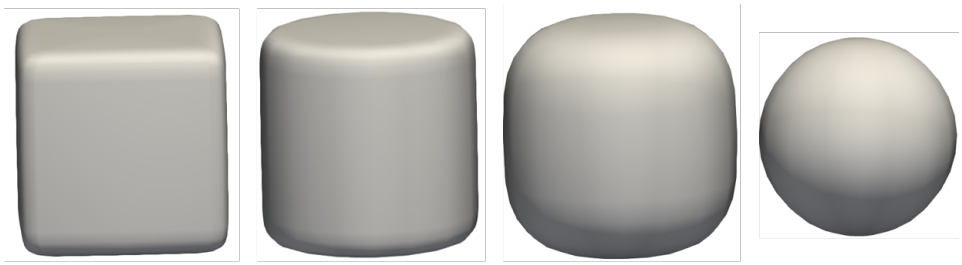

Figure 3.8: Different shapes of superquadric particles, based on the blockiness parameter.

where a, b, c are the half-lengths of the particles along their principle axes and $n1$ and $n2$ are blockiness parameters. Thus superquadric particles have five parameters which define the shape of the particle and lead to a balance between shape flexibility and model complexity. However, the usage of superquadric particles comes with the restriction of approximating solely ellipsoidal, box-like and cylinder-like objects. In Figure 3.8 several examples of different shapes, with respect to their blockiness are given. Furthermore, superquadrics particles are in contrast to simple spheres more computationally expensive due to the underlying contact model and the more complex contact detection [144]

Bonding Particles

As an enhancement to the classical DEM, the usage of bond forces in between at least two particles and the creation of bonded particles allow the simulation of liquid bridges, flexible behavior and possible breakages [150]. An example of a flexible conglomerate is shown in Figure 3.9, in which particles are boned in pairs to a bending beam [151]. The elastic bonds are connected in between two spheres, where the bond is fixed to the centers of the two spheres, respectively. Due to that the elastic bond rotates and translates along with the movement of the spheres, constituting to the deformation of the bonds and the particle itself. Based on [152] the forces and moments in normal and tangential direction acting on the bond are given by

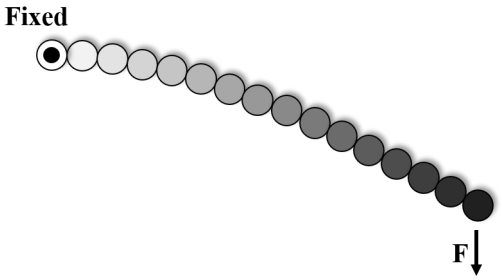

Figure 3.9: Simulated beam with modeled bonds [151].

$$F_{\mathrm{b,n}} = \sum_{\forall i} \delta F'_{\mathrm{b,n},i} + 2\beta_{\mathrm{b}} \sqrt{M_{\mathrm{e}} A_{\mathrm{b}} K_{\mathrm{n}} v_{\mathrm{n}}}, \tag{3.49}$$

$$F_{\mathrm{b,tg}} = \sum_{\forall i} \delta F'_{\mathrm{b,tg},i} + 2\beta_{\mathrm{b}} \sqrt{M_{\mathrm{e}} A_{\mathrm{b}} K_{\mathrm{tg}} v_{\mathrm{tg}}}, \tag{3.50}$$

$$M_{\mathrm{b,n}} = \sum_{\forall i} \delta M'_{\mathrm{b,n},i} + 2\beta_{\mathrm{b}} \sqrt{J_{\mathrm{s}} K_{\mathrm{tg}} I_{\mathrm{p}} \omega_n}, \tag{3.51}$$

$$M_{\mathrm{b,tg}} = \sum_{\forall i} \delta M'_{\mathrm{b,tg},i} + 2\beta_{\mathrm{b}} \sqrt{J_{\mathrm{s}} K_{\mathrm{n}} I \omega_{\mathrm{tg}}}, \tag{3.52}$$

where $F_{\mathrm{b,n}}, F_{\mathrm{b,tg}}$ are the normal and tangential bond forces while $M_{\mathrm{b,n}}, M_{\mathrm{b,tg}}$ describe the normal and tangential bond moments, respectively. In addition, the incremental forces and incremental moments caused by the linear spring are calculated by

$$\delta F'_{\mathrm{b,n},i} = K_{\mathrm{n}} A_{\mathrm{b}} v_{\mathrm{n}} \Delta t, \tag{3.53}$$

$$\delta F'_{\mathrm{b,tg},i} = K_{\mathrm{tg}} A_{\mathrm{b}} v_{\mathrm{tg}} \Delta t, \tag{3.54}$$

$$\delta M'_{\mathrm{b,n},i} = K_{\mathrm{tg}} I_{\mathrm{p}} \omega_{\mathrm{n}} \Delta t, \tag{3.55}$$

$$\delta M'_{\mathrm{b,tg}} = K_{\mathrm{n}} I \omega_{\mathrm{tg}} \Delta t, \tag{3.56}$$

where $\delta F'_{\mathrm{b,n},i}, \delta F'_{\mathrm{b,tg},i}$ are the incremental normal and tangential force and $\delta M'_{\mathrm{b,n},i}$ and $\delta M'_{\mathrm{b,tg}}$ are the torsional and bending moment calculated with the time step Δt [153], [154]. $K_{\mathrm{n}}, K_{\mathrm{tg}}$ are the normal and tangential bond stiffness constants, A_{b} is the cross section of the bond area and β_{b} is the local bond damping coefficient. M_{e} and J_{s} describe the mass and moment of inertia of the corresponding particles, whereas $v_{\mathrm{n}}, v_{\mathrm{tg}}$ and $\omega_{\mathrm{n}}, \omega_{\mathrm{tg}}$ are the normal and tangential translational and rotational velocity between these. I and I_{p} are the second area moment and polar area moment of inertia, respectively [152]. The two stiffness constants which are mainly responsible for bond behavior are determined by

$$K_{\mathrm{n}} = \frac{Y}{l_{\mathrm{b}}}, \tag{3.57}$$

$$K_{\mathrm{tg}} = \frac{K_{\mathrm{n}}}{2(1-\nu)}, \tag{3.58}$$

in which Y is the bond Young's module, ν is the Poisson's ratio and l_{b} the equilibrium bond length [152].

3.4.3 DEM Software Tool

Since the computational power increased and appropriate software tools and calibration procedures ensured qualitative and valid simulations, the DEM has become well established in academics and industry [126], [155]. There are many DEM tools available for 2D, as well as 3D simulations with different scopes, with either CPU or GPU computation abilities. Commercial software tools like EDEM [58], PFC [139] or Rocky-DEM [156] are often more intuitive and provide professional support. Contributed open-source software tools like Yade [157] or LIGGGHTS [142] allow a broad community of researchers working with the DEM and develop e.g. new applications or coupling methods.

In this work, the DEM software tool LIGGGHTS(R)-PUBLIC v3.8 of the DCS GmbH [142] is used which is provided as open-source and under the terms of the GNU General Public License. LIGGGHTS (LS) is an abbreviation for "LAMMPS improved for General Granular and Granular Heat Transfer Simulations" and based on the software for molecular dynamics LAMMPS (Large-Scale Atomic/Molecular Massively Parallel Simulator) [158]. Since LS has been implemented on a multi-processor basis, simulations of several million particles in an appropriate time were made possible [159].

LS is a text-based software tool and allows to call *compute* commands to calculate specific simulation parameters or *fix* commands to for example move geometries in the simulation domain. By importing standard STL files of the desired geometries, LS enables the rapid development of DEM simulations. The available source code is written in C++ and the knowledge transfer within the community allows modifying and adapting the software to the special needs of the individual developers. By changing the source code of LS, new contact models, specific geometry movements, new shapes, or bonding particles are implementable. The post-processing of the obtained DEM results is implemented with third party animation tools such as Paraview [160], where 3D animations or relevant plots are created to analyze the simulation behavior.

4

Machine Control of the PSM

In this chapter the developed methodologies are applied to the first RL application. It is the machine control of the novel PSM. In this chapter the principle idea, the development and digital twin of the PSM are described. Additionally, two parcel transportation tasks demonstrate the applicability of the developed methodologies and the combination of DEM simulation models as digital twins with RL. It is shown that the control of complex and high-dimensional DEM-based digital twins is made possible with the help of the developed methodologies of increasing the training speed and reducing the complexity. Finally, to create the possibility to use the developed RL frameworks for the real machine, a RL-PLC implementation is developed and presented exemplary.

4.1 Peristaltic Sortation Machine

The Peristaltic Singulation Machine (PSM) is mainly developed for the needs of the Courier Express Parcel (CEP) industry. However, the principle design and locomotion of the machine are also transferable to other goods and different industries. Over the last year the CEP industry underwent a constant growth especially fueled by the rising demand in E-commerce. In Germany alone, 83 % of 2.95 billion items were delivered as parcels in 2015 [161].The increasing number of parcels overwhelms the capacities of the participating industry partners and calls for higher performance especially in the distribution centres. One bottleneck in the distribution centres is the in-feed area where the discharged bulk parcel flow needs to be singulated i.e. each parcel is uniformly separated, spaced and aligned before entering the downstream sortation system [162]. Typically human unload operators singulate the bulk flow into a one-dimensional flow of

separated parcels with a throughput of approx. $850 - 900$ pph (parcels per hour).

To increase the performance throughput and get rid of manual labor, a special machinery is developed. These so called singulators, mainly based on conventional roller and belt conveyor technologies, as closed or open-loop systems, are commercially available and offered by at least three providers. Beumer's automatic parcel singulator consists of a combination of belt conveyors operated at different speeds and aligned rollers and is indicated with a throughput of up to $3,500$ pph [163]. The singulator Visicon by Siemens detects parcels with a camera system and yields to a throughput of $12,000$ pph with help of small belt conveyor units [164]. The Visicon singulator impresses with its high throughput but is only suitable for 2D bulk singulation. A three-dimensional bulk singulation is achieved by the use of inclined conveyor belts by Fives Accord and outputs up to $6,000$ pph, but is built as an open-loop system [165].

The developed PSM presents a system concept of a closed-loop singulator which is supposed to be able to separate two-dimensional and three-dimensional bulks into a single stream of parcels. It combines the functionalities of singulation, sortation and transportation of parcels in a single machine. Additionally, the used components and the peristaltic principle itself lead to a gentle and smooth transport of parcels compared to conventional systems.

4.1.1 Peristaltic Principle

The peristaltic or peristalsis is known as the flow generation by the propagation of waves along flexible walls [166]. Peristaltic action is an inherent neuromuscular property of any tubular smooth muscle structure e.g. in the human gastrointestinal tract or the locomotion of earthworms [167]. Inspired by this bionic principle, technical systems reproducing peristaltic movements are frequently developed and used. For example developments of peristaltic pumps are derived from this principle. These pumps are widely used with different designs and for a broad field of varying fluids [168], [169]. Furthermore, the adaption of the peristaltic principle enables new types of moving or crawling devices in the field of robotics [170], [171].

To use the peristalsis for transportation and singulation tasks, or particularly to manipulate the position of parcels, the principle has to be transferred onto a planar surface. In [172]–[175] a machine design which manipulates objects in several degrees of freedom while adapting the locomotion of the caterpillar into a XY-table has been developed. There, a closed loop-system, driven by inflatable air-chambers, manipulates objects at low speeds. In [176] air chamber actuators were exchanged by a mechatronic system to further improve the performance of the system. A conveyance table for roundish objects using linear actuation units and generic control algorithms has been developed by [177]. A manipulation by pneumatic actuators is reflected in [178] or by using three-dimensional displays in [179], [180]. To tackle the fault-tolerant control problem, [181] uses a flexible transport system based on linear actuators as well. A promising but

only theoretical work about an open-loop system demonstrates the usage of peristaltic movement to singulate parcels for the CEP-industry [182].

4.1.2 Development of the PSM

Following the idea of a horizontal sorting table and using peristaltic waves for singulation and transportation of parcels, requirements and design ideas for the peristaltic sortation machine have been made [183].

- The PSM must meet the requirements of the CEP industry and be able to transport, singulate and sort parcels of a certain size and weight using peristaltic movements.
- The system design shall be completely modular so that the system is scalable for the desired application. This scalability shall be achieved by adding or removing modules, scaling the size of individual parts and adjusting the number of actuators per module.
- To realize the peristaltic movements, multiple actuators shall be installed. By using moving instead of static actuators the maximum number of required units in the system shall be reduced.

Considering these requirements, a completely new construction of a peristaltic sortation machine has been designed. The machine design is distinguished into two individual parts, the actuation system with multiple actuator-units and the carrying system consisting of a flexible and resistant transport film. The actuation system is located beneath the flexible transport film and is able to manipulate the shape and structure of it. This manipulation is used to create certain moving waves with the film. By changing the height and speed of these waves or performing distinct motion patterns, the parcels are transported and singulated as shown in Figure 4.1.

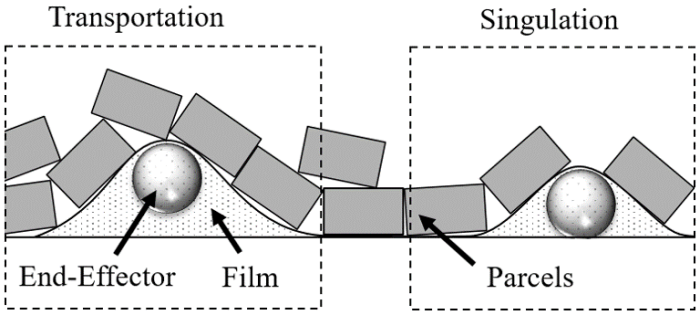

Figure 4.1: Side view of the peristaltic principle with transportation and singulation functionality.

To adapt the machine to the desired throughput and the special needs of the CEP industry, the size of the whole system is modifiable. Therefore, the design of the PSM is based on a

modular basis, where the actuation-system consisting of the table-unit which carries multiple actuator-units is extendable lengthwise. By adding multiple table-units, the width of the system is changeable, too. In this work one table-unit with a maximum two actuator-units and the flexible transport film is considered. This setup, depicted in Figure 4.2, can be loaded with multiple parcels and the machine is the able to perform singulation and transportation tasks simultaneously. The flexible transport film has a length of $4,500$ mm, a width of $2,000$ mm, is made of 1 mm PVC and therefore returns in its original state after manipulation. The contrast between the black material of the film and the color of common parcels allows a camera system to clearly identify parcels on the transport film. This camera system, consisting of three Cognex Insight 9012 cameras detects single and bulky parcels, transmits their position to the high-level PLC and is also able to read the barcodes of the parcels.

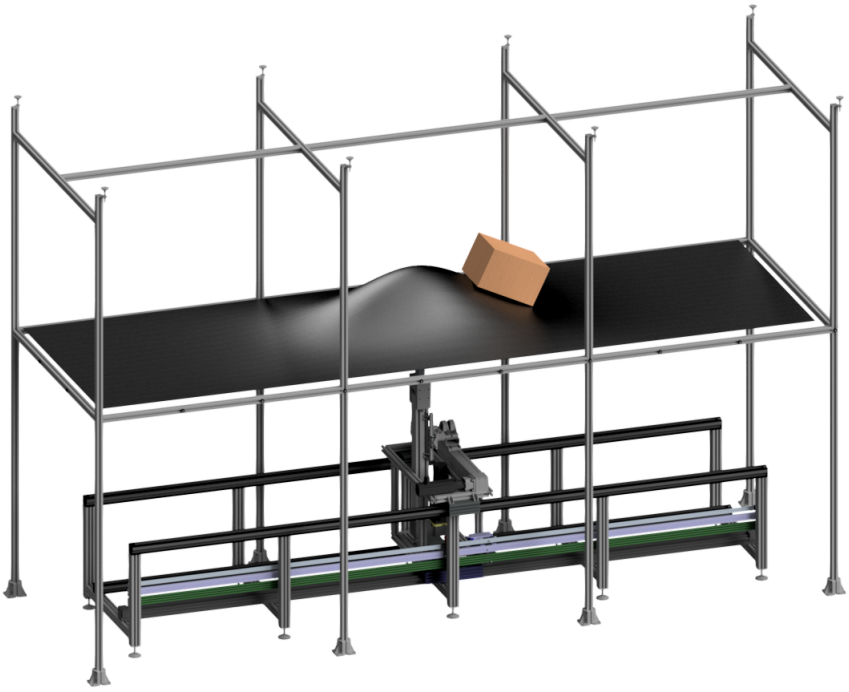

Figure 4.2: Design of the Peristaltic Sortation Machine.

The mechanical design of one actuator-unit is shown in Figure 4.3 and allows the unit to move in three different directions. The slide servo drive (2) equipped with a rack and pinion linear unit moves the whole unit along the X-axis across the table (1). To move in the YZ-plane, the unit has a tilting servo drive (3) with a linear unit which tilts the lifting drive (4). This lifting linear drive, made with a ball screw drive, can lift the end-effector (5) up to 550 mm. The two freewheeling and auto-align wheels of the end-effector are the only parts of the PSM which are in direct contact with the transport film and ensure a low friction. Considering the expected load and the mass of the entire system, the controllers of the individual servo drives are adjusted to ensure a constant acceleration and stable velocity. All drives are equipped with appropriate gears

to apply the necessary torque. These configurations generate individual dynamics for each axis which are shortly called slide, tilt and lift axis. The slide and lift axes move translational while the tilt axis moves rotatory, caused by the joint on top of the linear unit. The resulting dynamics of the axes are described in Table 4.1.

Table 4.1: Resulting dynamics of the axes of the actuator-unit.

Axis	**Acceleration**	**Max. Velocity**
Slide	1,000 $\mathrm{mm/s^2}$	1,000 $\mathrm{mm/s}$
Tilt	0.9 $\mathrm{rad/s^2}$	0.36 $\mathrm{rad/s}$
Lift	500 $\mathrm{mm/s^2}$	360 $\mathrm{mm/s}$

The entire machine is equipped with a state-of-the-art PLC. This high-level PLC, a Siemens 427e IPC, provides a Human-Machine-Interface (HMI), receives the current status of the machine with a variety of sensor equipment and controls the position of the individual actuator units. It obtains information about the absolute position of the actuator-units, gathers information about the parcels on the transport film with the camera system and activates certain singulation and transportation tasks which are performed with the help of suitable algorithms.

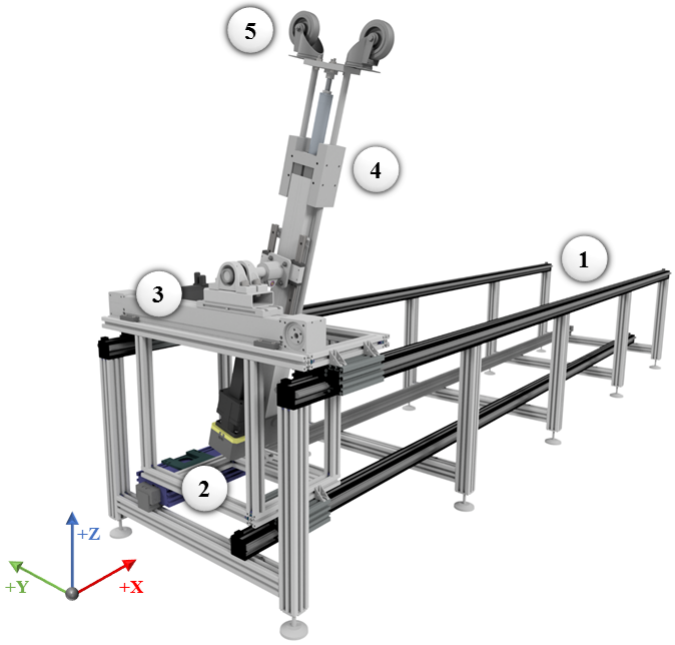

Figure 4.3: Mechanical design of the actuator-unit.

4.2 Digital Twin of the PSM

As already stated in Section 3.3, a digital twin is the virtual replica of the desired process or product. Depending on the current development phase, ranging from designing to service and after-sales, the content and complexity of the digital twin can vary. During the development of the PSM, the digital twin was essentially and especially required in the early development phase even before a mechanical or a prototype was designed. Furthermore, for a suitable machine control, the digital twin is necessary to model the real behavior of the machine parts and the parcels.

The purpose of the digital twin of the PSM is to replace the real machine and use it as a surrogate model of the physical PSM to develop and train RL approaches for different machine control tasks. Changing of individual components or parameters and observations of the general behavior of the actuator-units, transport film and parcels are obtainable in the simulation without the costs for real hardware or operators. Besides, in episodic scenarios like in RL, the training can run $24/7$ with several parallel instances and the environments can be reset automatically. However, it must be ensured that the model represents the real PSM within certain limits. In [184], the parcel transportation behavior of the simulation and real machine were sufficiently tested and validated. The measurable dynamics of the actuator-units are properly validated while the behavior of the flexible film and the parcels dynamics were empirically tuned and tested. Due to the fact, that the machine is still in the prototyping phase and the design is not yet finished, the simulation might differ but nevertheless assists to develop machine control algorithms.

The inherent complexity of this machine, due to the interaction between moving mechanical parts of the actuator-units, the intrinsic behavior of the flexible transport film and the rigid body dynamics of moving parcels, has to be modeled accordingly. While the motion of the actuator-units can be described analytically with laws of motion, the composition of these many different objects leads to an analytical consideration that is hardly feasible. The moving peristaltic waves which manipulate various parcels in a bulky condition and occurring mutual collisions of these are difficult to treat analytically and require highly complex simulation models [185]. That is why the DEM, essentially made for handling colliding objects, is here used as a promising tool to simulate most aspects of the PSM in this work. This DEM-based digital twin then needs to be coupled with an appropriate RL framework.

The usability of standard machine learning libraries such as PyTorch [186] is made possible by embedding the DEM into a Python 3 framework as described in Section 4.3.1. The data-driven DEM model is therefore integrated as a dynamic environment in the RL context. Due to that suitable interfaces to exchange observations or actions as shown in Figure 4.4 are defined. The required RL agent parameters like the agent structure, learning rate, etc. as well as the necessary simulation parameters such as the time step, simulation, number of parcels, etc. are directly in Python adjustable.

Considering the previous explanations of the DEM, this section outlines the individual parts of the digital twin of the PSM and its realization in the simulation domain. The PSM is analyzed and divided into the mechanical parts, the approximation of parcels and the flexible transport film.

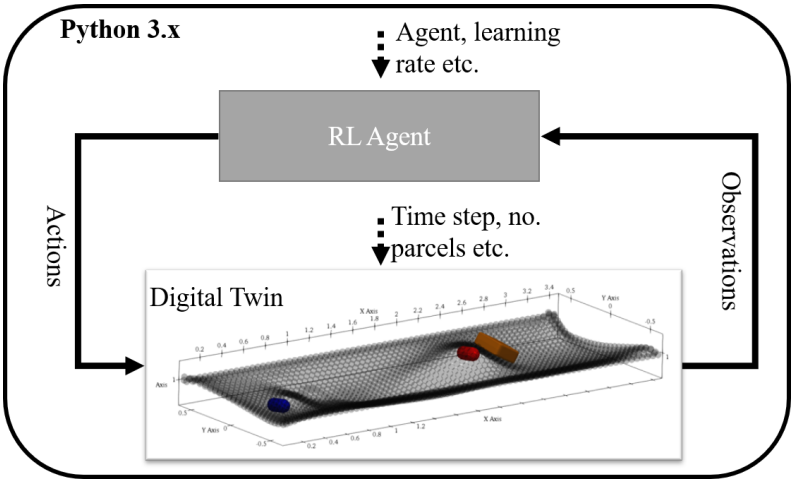

Figure 4.4: Digital Twin framework of the PSM.

4.2.1 Mechanical Parts

The DEM originates from the field of particle and molecule simulations, but is also able to simulate static or moving objects which are in contact with the researched material. These objects, also called geometries, can be directly imported from the original CAD in STL-format. However, only the parts which could be in direct contact with the desired material are modeled, whereas the other parts are neglected. Usually, only the surfaces of objects are approximated, imported in the DEM and freely movable inside the simulation domain. Then they can be then moved linear, rotational are even be oscillated. Upcoming forces trough material impacts or possible wearing of the geometries is detectable with the node-wise resolution of the STL-file [142]. Importing of original CAD files, especially in the beginning of the development phase, supports the use of digital twins with the DEM as a tool for rapid prototyping. Even before the new machine or plant is designed, individual parts can be tested fast and analyzed e.g. regarding expected velocities or possible wearing [128]. Potential redesigns in an early stage of the development phase are thus easily adaptable and can be reevaluated. However, to simulate acting forces on geometries and model the resulting counteractions, coupled multi-body simulations are required [57].

With regard to the PSM design, only the end-effectors which are at the very end of each actuator-unit are of interest for the DEM simulation. Figure 4.5 shows the end-effector in CAD as well as in the simplified STL-file format. The kinematic of the two freewheeling and auto-aligned wheels is replaced by an ellipsoid representing the actual radius of action.

Moreover, a simplified conversion with a small number of nodes fastens the computation and barely influences the simulation results. However, the complete kinematics of the actuator-units have to be simulated as well. Derived from the mechanical design and the performance of the electrical drives and gears, the trajectories of the axes are implemented. Especially the limits of the system, the maximum velocities and acceleration ramps are calculated in the high-level programming language Python and the transferred to the DEM simulation, which is explained in Section 4.3.1. The derived kinematics, dependencies between the three axes and velocity profiles are explained later in Section 4.4.1.

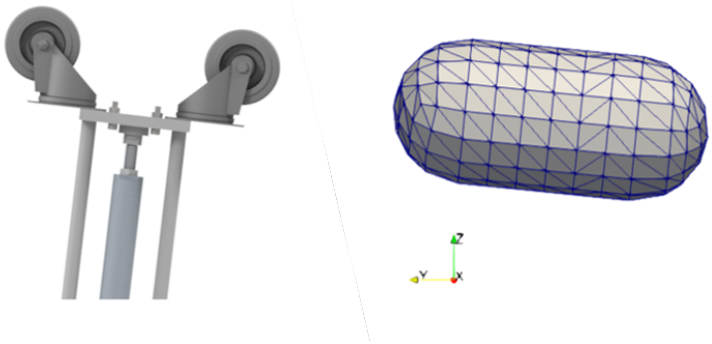

Figure 4.5: End-Effector, left: Original CAD, right: Simplified STL.

4.2.2 Parcel Approximation

As already mentioned in Section 3.4.2, the classical DEM uses spherical particles to simulate real world behavior. Roundish but not completely spherical particles can be simulated by adapting the coefficient of rolling-friction (CORF) to mimicry a shape-like behavior. Apart from this, more complex shapes are approximated by e.g. the multi-sphere or superquadric technique.

It is assumed that parcels or packages mostly have a cuboid shape, therefore Figure 4.6 shows the approximation with both approaches. To sufficiently represent a cuboid using the multi-sphere approach a large number of particles is required. Even though the computation time therefore theoretically increases, the computation of the superquadric approach is not substantially faster. The more complex and time-costly contact detection of superquadrics puts the speed savings into perspective. But, as with all multi-sphere approximations, the interlocking problem, which is a mutual locking of shapes with their microstructures, significantly affects the interaction of the bodies [187]. In summary, the approximation of parcels in the DEM using the superquadric approach, without the presence of sharp edges, yields to sufficient results and has also been tested with standard conveyor belts [133]. To adjust superquadrics to represent cuboid particles, the blockiness parameters are set to $n_1 = n_2 = 10$. It should be noted that, considering the processing of parcels in distribution centers, information about the weight of parcels is typically neither measured nor stored as data on the parcel as a bar-code, etc. Moreover,

the exact position of the center of gravity in the local coordinate system of the parcel is not given. Therefore, it is assumed that the center of gravity is set at the point of origin of the parcel and the weight of the parcel individually defined.

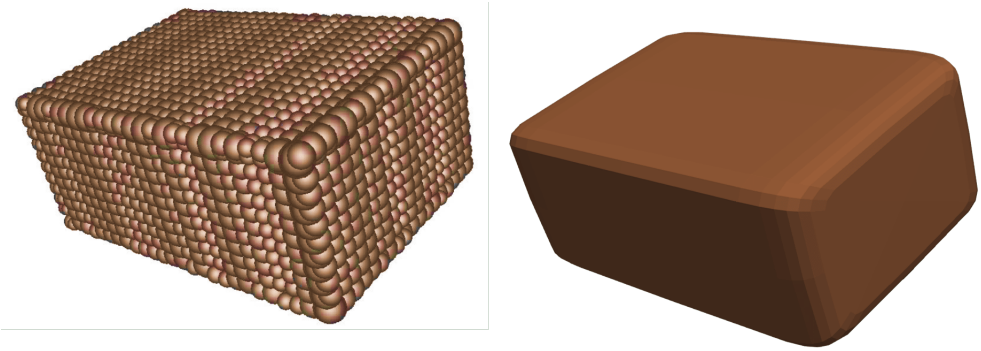

Figure 4.6: Parcel approximation, left: Multi-sphere, right: Superquadric.

For a suitable representation of the behavior of the parcels, the contact model of the DEM has to be calibrated in addition to the shape approximation and the size of the parcels. The behavior of the parcels is tuned to realistically perform on the flexible transport film as well as interacting with other parcels. The calibration of parcels differs from commonly applied DEM calibration procedures [183]. Apart from the Young's and shear modulus, the coefficient of friction (COF) and the coefficient of restitution (COR) fully describe the contact behavior in the simulation. Here, the value of the coefficient of friction can be directly measured with the inclined plane procedure. The COF can be calculated with

$$\mu_\mathrm{r} = \arctan(\alpha_\mathrm{r}), \tag{4.1}$$

where α_r is the measured angle with the inclined plane at which the parcel starts moving downwards.

The approximation of parcels in the DEM requires the determination of a suitable contact model and calibrated material parameters. The used contact model for this simulation is defined by the Hertz-Mindlin tangential history model with rolling friction described with the *epsd2* model and adjusted for superquadrics. The atom style is set to hybrid-superquadric, with one type of bonds and five bonds per atom. All material parameters are shown in Table 4.2. Due to the fact that parcels can not be calibrated by standard calibration scenarios, e.g. the AoR tests, the material values are calibrated empirically. Therefore, the simulation results are compared to the behavior of the real machine interacting with parcels and continuously updated or tuned, respectively. Additionally, the results underline the correctness of the simulation and validate the material parameters. In [184] the behavior of parcels, varying in size, weight and surface roughness, was analyzed and the settings for a suitable parcel transport with the PSM were determined. A successful parcel transport dependents on the height and speed of the

transportation wave as well as on the weight of the parcels. As seen in Figure 4.7, the height of the wave creates a resulting angle α. When reaching a specific angle the parcel overwhelms the static friction to the transport film and surfs downwards or is transported in forward direction, respectively. Since heavier parcels are pushed deeper into the transport film, the resulting angle changes. Additionally, the height of the transport wave is adjusted individually considering the inertia of the parcels and to achieve a desired transport velocity.

Table 4.2: PSM simulation parameters.

Property	**Symbol**	**Unit**	**Value**
Time step	Δt	s	5e-5
Young's module	Y	Pa	5e6
Poisson's ratio	ν	-	0.3
COR	e	-	0.055
COF (parcel-parcel)	μ_{pp}	-	0.02
COF (parcel-effector)	μ_{pe}	-	0.02
COF (parcel-film)	μ_{pf}	-	0.6
CORF (parcel-parcel)	$\mu_{r,\mathrm{pp}}$	-	0.01
CORF (parcel-effector)	$\mu_{r,\mathrm{pe}}$	-	0.05
CORF (parcel-film)	$\mu_{r,\mathrm{pf}}$	-	0.9

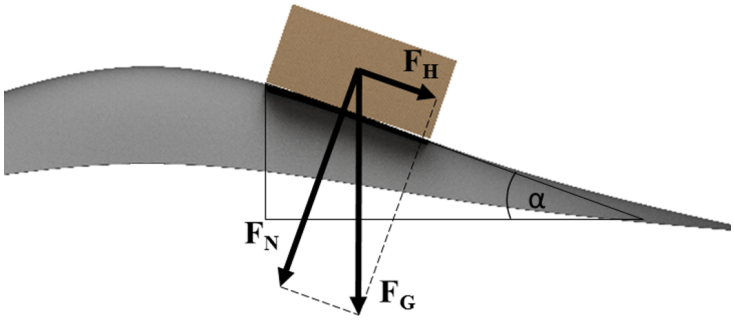

Figure 4.7: Parcel transportation mechanism.

4.2.3 Flexible Transport Film

The flexible transport film of the real machine is made of 1 mm PVC and requires a suitable modelling of its flexible behavior and a simulation of its interaction with the end-effectors and the parcels. Flexibility in terms of the DEM, apart from rigid body simulation, can be achieved by combining particles with a flexible conglomerate. In contrast to the multi-sphere approach, where particles are fixed in a local coordinate system, the combination with the bond functionality forms a wholly flexible body. Thus, the particles can move freely in the conglomerate, but are

limited by the applied bond forces. The possibility of a bond breakage can be activated and set with suitable values, too [152].

It should be noted that this is the first time that the DEM is used to simulate a partially fixed and flexible PVC film, interacting with geometries and other rigid bodies. The simulation of the flexible transport film is formed with one conglomerate consisting of one layer of particles which are bonded together. Figure 4.8 depicts the film in the DEM simulation. As shown, the film is approximated with spherical particles, which are actual superquadrics, which blockiness parameters $n_1 = n_2 = 2$ are adjusted to be spherical. Tests with other shapes caused interlocking problems and internal overlapping of the film. The diameter of the individual spheres and thus the total number of spheres within the flexible film is a trade-off between the accuracy of the approximation and the required computation time. Spheres with a diameter of 5 mm yield to an acceptable simulation time and show sufficient results. Admittedly, smaller particles could achieve more accurate results. But, especially with regard to the deploying the DEM as a reinforcement learning environment which needs to be repeatedly simulated hundreds of times, a time saving approximation is desired.

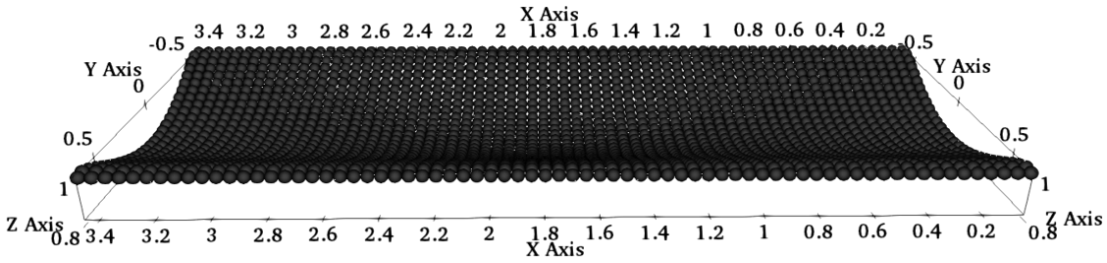

Figure 4.8: Simulation of the flexible transport film.

As there is no standardized calibration procedure for DEM particles with bond forces established and this is the first time of mimicking the behavior of a PVC transport film with the DEM, an empirical calibration is conducted. Usually, bonding particles freely move through the simulation domain. In this simulation through the flexible transport film is kept at a distinct height above the actuator-unit and has to be able to carry the load of multiple parcels. Thus, the outer particles in X-direction are frozen and unable to change their position. Each particle of the film is bounded to a maximum of 5 neighbouring particles and the bond parameters are tuned. The simulation of the general behavior and flexibility of the PVC film achieved a high degree of reality with the parameters shown in Table 4.3.

Especially the initialization phase of the film in the simulation is a tedious task. First, a lattice consisting of one layer of particles is generated and the bonds between the relevant particles are activated. Then the edges of the film are slightly moved together without the influence of gravity. After that, the relevant particles at the edges are frozen. With incipient gravity, the film stretches and it reaches its idle state after a certain settling time. This entire initialization ensures

a realistic model of the film but requires additional computation time at the beginning of each simulation.

The length and width of the film are accordingly changeable. In this work, the edges in Y-direction are not frozen, but open-ended. In contrast to the real-machine where all edges are mounted, in the simulation only a section of the film is considered. A mounting all-around or freezing respectively would cause reflection waves. Therefore, as seen in Figure 4.8, both ends are simulated as loose ends to avoid unrealistic reflections. Changes of the length and width of the film as well as the weight of the film might require adjustments of the bond parameter because of the internal strength of the film and bonds, respectively.

Table 4.3: Bond parameters of flexible transport film.

Property	**Symbols**	**Unit**	**Value**
Particle diameter	d	mm	5.0
Bond length diameter	d	mm	5.0
Bond Young's module	Y	Pa	4e5
Bond Shear modulus	G	Pa	1e5

4.2.4 PSM Simulation

The formerly described individual components are composed into one DEM simulation. A qualitative comparison of the simulation with the real PSM validates the simulation parameters and allows to use of the PSM simulation as a digital twin. This simulation is integrable into a RL framework and properly usable as an environment. As described in Section 4.3.1 the simulation can be instantiated for multiple environments with distinct pre-parameters. Depending on the actual task, the size of the transport film, the number of actuator-units or the number and kind of the parcels are adjustable. Figure 4.9 shows an entire environment with two actuator-units and one single parcel. This simulation innovatively combines superquadric and bonded particles within one simulation domain and the mechanical drive control relocated in Python.

According to the described coupling method, this simulation is callable in Python, the digital twin is applicable as an environment and easily adjustable with a few hyperparameters. Regarding the dynamics of the electrical drives and linear units, the acceleration, velocity and position of the actuator-units are changeable according to the desired action-space. To computationally observe the PSM simulation, the parcel properties like the position, velocity, angular velocity, etc. are directly analyzable within the Python framework. Additionally, an exceeding of the machine limits and impossible parcel movements are automatically handled and considered

The full integration into Python results in a fast and easy implementation of DEM environments into an object-oriented architecture. There, standardized properties and pre-parameters

are easily changeable and the coupling facilitates the call of several parallel instances to speed up the training process. The post-processing of the results is either conducted directly in Python i.e. observing the gathered rewards per episode or via Paraview to visualize the PSM simulation and generate three-dimensional animations.

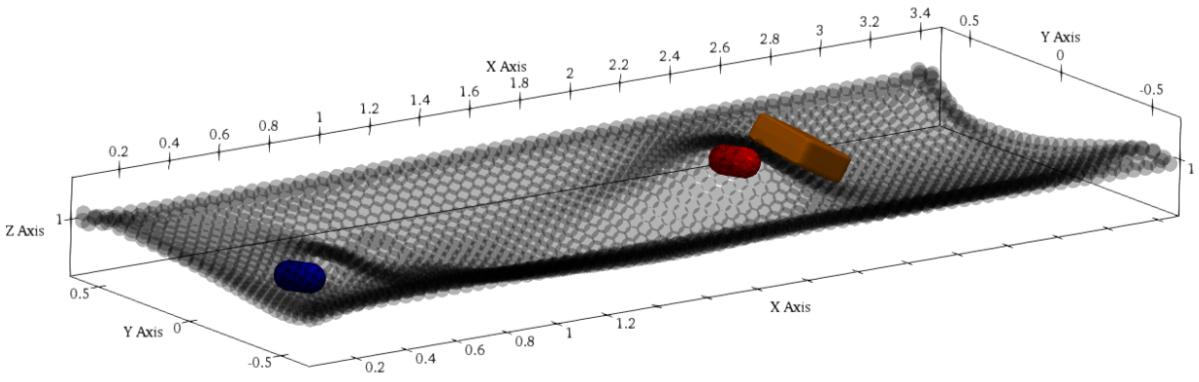

Figure 4.9: Digital twin of the PSM as environment.

4.3 Methodology

Running DEM based digital twins together with reinforcement learning algorithms requires certain methodologies to couple DEM simulations with machine learning and handle the computationally slow but very complex DEM simulation models. Therefore, as a first methodology, coupling the DEM describes a novel way of the integration of DEM simulations into RL frameworks. To successfully enable the agent to handle complex and high dimensional tasks, the iterative learning schedule is developed. This schedule is able to manage multi-complex environments with increasing complexity, expedite the entire training and compute suitable network structures of the agent. Additionally, the developed distributed ACRL methodology no only allows to train an agent with multiple instances of the same environment to speed up the training significantly, but also to properly use DEM based digital twins as RL environments. Finally, by splitting up very complex and extensive tasks into smaller sub-tasks, an adapted hierarchical RL approach is developed which perfectly suits to DEM environments by activating predefined deterministic sub-controls.

4.3.1 Coupling the DEM

To allow to use the DEM in a data-based or machine learning context, it is required to develop a suitable coupling framework is developed. Therefore, the coupling methodology for the DEM is developed to achieve two functionalities. First, the results of DEM simulation models are supposed to be exchangeable with and evaluable by RL agents. Second, the DEM simulation

shall be enhanced, from originally self-contained, to be event-based and sequentially executable. The coupling shall also enable usage of different DEM simulation models as environments which dynamically interact with RL agents.

To be able to use RL in combination with DEM simulation models, as described in Section 2.2, a program based with an event or episode wise bidirectional coupling needs to be established. Embedding the DEM into a broadly used high-level programming language allows to link the simulation details and results to other applications. Hence, the programming language acts as a coupling element and allows the exchange of data between e.g. RL algorithms and the DEM simulations. Therefore, the co-simulation framework describing the coupling of DEM simulation models with RL algorithms is setup with an embedding of LS into Python 3.

Python is one of the most popular programming languages in science which imported a large number of contributed and open-source libraries, especially in the field of machine learning [188]. The interpreter language Python is not the most powerful programming language but nevertheless results in fast computation while using the extension for tensor-based calculations or highly parallel computation with GPUs e.g. using CUDA [189]. To facilitate the use of machine learning and allow fast algorithm developments, researchers use contributed libraries containing relevant functionalities and documentations. One promising library, also for RL, is Pytorch which has the advantages of a very logical structure and appropriate performance and was mainly boosted by OpenAI [190].

In this work the DEM tool LIGGGHTS(R)-PUBLIC v3.8 (LS) is used which does not inherently support the integrated usage of the required machine learning techniques, but can be coupled to other applications over the network or program based. The standard LS inherently supports the possibility of a coupling with Python 2 which is however discontinued in the year 2020. In this work, the coupling is therefore enhanced by changing the provided LS open-source code to be able to couple with Python 3 and preferably use state-of-the-art tools and libraries. In [191] a coupling between LS and Python 3 is developed and focused on an object-oriented architecture. There, multiple LS instances are called and the final results post-processed via standard python libraries. But to use the DEM as an environment in a RL context, interim results as states and action commands affecting the simulation during run-time, have to be bidirectionally exchanged.

The developed methodology of the DEM-Python coupling is shown in Figure 4.10. LS is wrapped as an external library and can be called as a Python LS object. The mainframe calls the objects with individual or similar pre-parameters which define e.g. simulation properties, material parameters, etc. In the depicted case, three LS objects are called, where two of them are initialized with the same parameters. The LS objects are either called only on one or decentralized on n-different threads of the CPU. This approach is only constrained by the computational power of the used machine and generally adaptable to any number of LS objects.

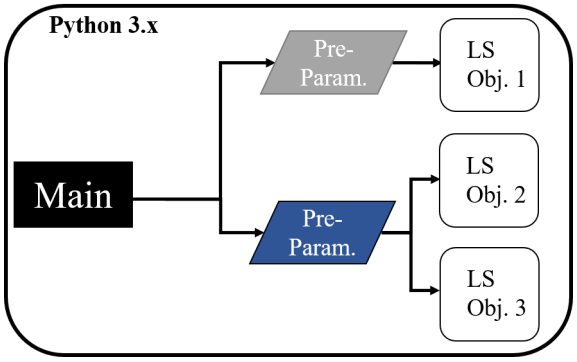

Figure 4.10: DEM-Python coupling structure.

The bidirectional communication between the mainframe and LS is achieved by executing typical LS commands and reading particle-wise data like positions or forces. With these commands simulation properties can be exchanged, particles inserted, or geometries moved in the simulation domain.

Typically, DEM simulations are defined once, then executed and the results observed after the simulation is finished. The developed coupling methodology however also allows to run the DEM simulation sequentially. As a result, the DEM simulation automatically stops after the last execution, the main program observes the current results and adds certain commands to continue the simulation. Therefore, this method offers event-based and dynamic interaction with the DEM simulation which is required to use the DEM as a RL environment. Theoretically, data exchange is possible for every time step of the DEM and thus the main program could interrupt the simulation at any time. Although LS is wrapped into Python, the data exchange is computationally time-consuming so that a data-exchange in certain time-slots is foreseen. By coupling LS into a RL framework, actions with a discrete time step are performed while the data-exchange is executed afterwards.

A developed standard interface allows to use of the DEM-based environments in the Python framework and can exchange n-dimensional states, continuous or discrete actions. The reward signal is automatically forwarded to the Python RL agent. Furthermore, the developed approach to asynchronously updates the Python RL agent while one or multiple environments are executing actions.

Apart from applying machine learning libraries, the DEM-Python coupling also offers further advantages and extends the possibilities of LS. Especially while simulating machine motions and moving mechanical parts, LS reaches certain limits. It inherently does not support motion equations based on accelerations, but constant velocities over certain time steps. To move bodies with a constant acceleration, it is necessary to discretize the velocity profiles in Python which then results in small deviations to the ideal acceleration ramps. Additionally, the coupling offers the possibility to suitably handle the LS parameters, save simulation results, or visualize

important simulation characteristics. Regarding these advantages LS is solely executed and managed via Python in this work.

4.3.2 Iterative Learning Schedule

The iterative learning schedule as one type of a curriculum learning approach is developed to alternate between environments and tasks of increasing complexity. In specific, the learning is split into at least two stages using one simplified and one complex environment. By gradually increasing the complexity of desired tasks, the curriculum learning approach generally benefits by decreasing the speed of convergence and therefore of the entire training process [192]. These benefits are incorporated in the developed iterative learning schedule which is especially applied to complex tasks where a hyperparameter search with the desired environment is unfeasible due to too long computation times.

The iterative learning schedule is built as a two stage curriculum learning approach. As described in [193], curriculum learning is addressed to tasks that are too difficult to learn from scratch. Therefore, by following a certain *curriculum*, the accomplishment of tasks in ascending order is learned in RL using transfer learning. Gained knowledge and experience of previous leanings are transferred to leverage the training of the final task. Curriculum learning can be used supportively to successfully learn to accomplish a set of tasks $\mathcal{T}$ with varying MDPs e.g. in the state or action space, or the reward function.

More formally, the task i as an MDP is defined as

$$m_i = (\mathcal{S}_i, \mathcal{A}_i, P_i, r_i) \in \mathcal{T}, \tag{4.2}$$

where the state space of task i is defined as a subspace of the whole state space, i.e. $\mathcal{S}_i \subseteq \mathcal{S}$. In case of complex and very detailed environments, the state space is continuous and high-dimensional which leads to an intense complex RL task. To reduce the complexity and fasten the training, simplified environments with a reduced state space for the individual sub-tasks are derived from it. Similarly, the action space $\mathcal{A}_i \subseteq \mathcal{A}$ can be reduced to fasten the training, as well. By defining a subset of the state space $\mathcal{S}_i$ for the individual tasks i the transition probability $P(s_{t+1}|s_t = s, a_t = a)$ yields to a restriction to $\mathcal{S}_i$ with

$$P_i = P|_{\mathcal{S}_i}, \tag{4.3}$$

and a restriction of the reward function $r(s, a)$ to $\mathcal{S}_i$

$$r_i = r|_{\mathcal{S}_i}, \tag{4.4}$$

where the reward function might not be shared between different tasks. In the iterative learning schedule the agent is trained first with a small subset of the state space with a simplified

environment and the knowledge is transferred to the next complex stage. Therefore, the training of the entire main task is expedite by splitting up into smaller task with faster convergence. However, the dimensionality of the state space and the complete action space of the complex environment remain equal in the simplified environment. That is way the simplified environment forms an extract of the original state space and behaves consistently. The level of reduction depends on the actual environment and application and has to be set individually.

The developed iterative learning schedule assists the agent to learn highly complex tasks and also determines suitable hyperparameters of the used DNN in the RL algorithm with multi-complex environments. Within the schedule, the determination of the network structure of the DNN and the learning parameters are shifted to the training with simpler environments of much faster computation. Especially the network size and the depth of actor- and critic-networks of the RL agent have to be adjusted properly to ensure high-quality training results. It is known that the final structure of the DNN and the setting of the hyperparameters strongly affect the learning of the RL algorithm. Therefore, in [194], a vast grid search of all relevant hyperparameters has been conducted to investigate the major influences. There, the network size was recognized as an important parameter that needs to be set carefully for the individual task and environment. The *adam* optimizer [195] and *he_uniform* initialization [196] of the network weights appear to sufficiently perform for all tasks and are therefore used for all function approximators in this work.

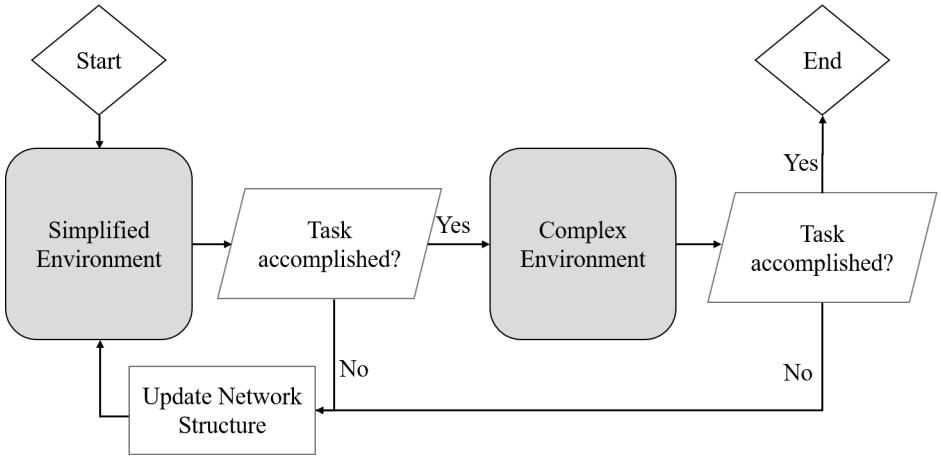

Figure 4.11: Framework of the iterative RL approach.

As depicted in the framework of the iterative learning RL approach illustrated in Figure 4.11, the agent is trained first with the simplified environment and a proper network structure is computed. Since the dimensionalities of the simplified and complex environment are equal, the computed network structure fits for the complex environments, too. The determination of the ideal size and structure of a feed-forward neural network is a crucial task. As the size depends on the desired task and the training data, a manual adaption of the network size is the norm.

Since very wide and deep neural networks consists of a huge set of trainable parameters, they aim to take long training times compared to small and shallow networks. Therefore, the width and depth of the desired network are ideally as small as possible. Two general approaches to adapt the width and depth of a DNN are denoted as growing and pruning [197]. Since small networks tend to underfitting and large networks to overfitting, an ideal network size is desired. In growing, a comparatively small network is trained and accordingly increased in width and depth to find an optimal network structure. Contrary to this, the pruning method starts with a relatively large network and decreases the network size by pruning individual neurons.

In this work, the growing method forms the basis of the network adapting procedure of the iterative learning schedule. These update rules are set by default, but the rules can be adjusted manually, e.g. for large networks. Considering a state space with $\mathbb{R}^d$ and the dimensionality d of the input space, the default network width starts with $n = d$ neurons per hidden layer m. The number of neurons per layer is kept constant in each layer. Considering an actor-critic RL agent with two separate networks for actor and critic, the default number of hidden layers for the actor is aimed to be $m_a = 2$ and for the critic $m_c = m_a - 1$. Note that these values are set arbitrary, but in the case of existing expert knowledge, the default values are adjustable to reduce the number of required iterations.

The iterative learning schedule in Figure 4.11 starts with the training of the agent with the simplified environment and the default network structure. If the training with the simplified environment is not successful and the task can not be accomplished until the maximum episodes are reached, the network structure will be adapted. The number of neurons per layer is automatically updated with $n = n + 2$ and the training with the simplified environment repeated. A hyperparameter then decides about the maximum number of neurons per layer and is set to $n_{max} = 128$ neurons per default. If this threshold is reached and the task can not be accomplished, the number of hidden layers will be increased with $m_a = m_a + 1$ and the number of neurons set in the new layer to the default value of $n = d$. The number of neurons in the previous layers remain. When the task is finally accomplished the iterative learning schedule remains with the current network structure and switches to the second stage with the complex environment automatically.

Similar to other curriculum learning approaches, the gained knowledge of the training with the simplified environment is transferred to the training with the complex environment. If the desired task can also be accomplished with the complex environment, the iterative learning schedule is finished and the RL agent is trained with a proper network structure. If the task can not be accomplished, the network structure is updated accordingly and the schedule is looped back with the simplified environment as depicted in the framework of the iterative RL approach.

4.3.3 Distributed ACRL

The complex DEM based digital twin of the PSM requires significant computational resources and thus enormous computation times. Parallelism helps to reduce the computation time in various applications and is also researched in the area of RL to decrease training times. In A3C for example, multiple parallel agents are trained with instances of the same environment contributing their knowledge to a global network[15]. There, each agent, called worker, interacts with its environment, calculates the values and policy losses and finally updates the global network individually. Thus, A3C helps to stabilize the training and finally reduces the entire training time [15].

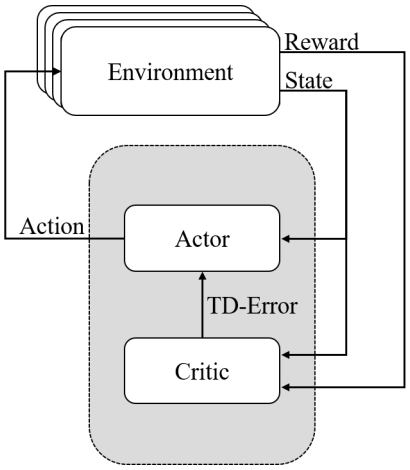

Figure 4.12: Structure of the distributed version of ACRL

Based on the A3C approach a novel distributed ACRL approach is developed which generally speeds up the exploration of the state space using one agent and multiple parallel environments. This parallelism, especially developed for slow solvers, also drastically reduces the required computation time of the entire training. The developed approach is shown in Figure 4.12 with one main ACRL agent connected to multiple distributed parallel environments. The number of parallel environments depends on the complexity of the desired tasks and the computational resources. The environments run asynchronously and the agent can alternate between them. It is important to note, that the agent has no information about the number or presence of the parallel environments. The k parallel environments run on different CPU threads of a single machine and allow an efficient parallel computation without communicating with external objects. However, parallelism through multiple machines is feasible, too. Every environment is initialized individually, receives actions a_t^k from the main agent based on the current policy and the observations s_t^k. Also, the agent receives rewards r_t^k which are individually assigned by the environments. In every step of one of the environments k the advantage function is calculated

with

$$A^{\pi_{\boldsymbol{\theta}}}(s_t^k, a_t^k) = r_t^k + \gamma V^{\pi}(s_{t+1}^k) - V^{\pi}(s_t^k), \tag{4.5}$$

and the policy parameter θ is updated.

The entire distributed ACRL algorithm is shown below. It is worth to note, that the one ACRL agent alternates between all the environments and is updated immediately after each step of every of the environments. The complete training is performed with a distinct number of total episodes for each environment. The episodes run either until the maximum number of steps T or a terminal state is reached.

Since the agent interacts with the environments asynchronously a certain timing is necessary. The timing considers the required time to update the policy t_π and the computation time of the environment to advance to the next step t_{cmp}. Typically, when dealing with computationally slow environments $t_\pi << t_{\text{cmp}}$, the timing is not crucial, but the agent has to hold a waiting position to receive new state information.

Algorithm 4 Distributed ACRL

Initialize ARCL agent
Run k parallel environments for ϵ episodes:
Environment k:
Randomly initialize environment k
while $t < T \;||\; s_t$ is terminal state **do**
Randomly initialize environment k
Take action with policy $\pi_{\boldsymbol{\theta}}$ and sample $s_t^k, a_t^k, s_{t+1}^k, r_t^k$
update V^{π} with $r + \gamma V^{\pi}(s_{t+1}^k)$
Calculate advantage $A^{\pi_{\boldsymbol{\theta}}}(s_t^k, a_t^k) = r_t^k + \gamma V^{\pi}(s_{t+1}^k) - V^{\pi}(s_t^k)$
Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx A^{\pi_{\boldsymbol{\theta}}}(s_t^k, a_t^k) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^k | (s_t^k)$
Update the ARCL agent $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
Advance to next step $t = t + 1$
end while

The distributed ACRL approach drastically increases the sampling rate compared to using only one single environment. To stabilize the training of a single agent asynchronously interacting with multiple environments, Experience Replay (ER) is introduced [95]. In ER a replay buffer is filled with the sampled experience of one or multiple environments. The agent replays the samples of the replay buffer, updates its policies accordingly and thus increases the sample efficiency [198]. ER also prevents overfitting and tends to converge faster during the training. Especially with computationally slow environments and thus relatively fewer data per time, ER allows to use the available data more efficiently. In this approach a replay buffer of a certain length n is filled with samples of the environments represented by their experience $e^k = s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}...$ using the First-In-First-Out (FIFO) principle.

This approach is applicable onto any computationally slow environment and can be easily integrated into other methodologies or different replay buffers. The strategy of the distributed ACRL approach in combination with an ER buffer lead to efficient training times, was successfully tested with comparatively slow DEM simulations [185] and is therefore used in the following of this work.

4.3.4 Hierarchical RL Framework

In RL an agent is trained to search for a policy that maximizes the cumulative reward. This tends to be challenging especially with large state spaces and long term horizons with standard learning approaches. Therefore, additionally to the iterative learning schedule, the developed HRL framework abstracts and reduces the complexity of difficult tasks to further decreases the convergence speed and thus the overall training time. Based on the idea of decomposing a complex task into smaller sub-tasks and solving these individually or simultaneously, the developed approach consists of a hierarchy pyramid which is shown in Figure 4.13. This approach involves two novelties namely the using of deterministic polices instead of learning agents and the distinct reduction of the state and action space per level.

Instead of using different managers as described in Section 3.2.6, the top level is performed by one single master-agent. This agent observes the environment of the specific problem entirely and monitors the lower levels to achieve the desired goal. Depending on the temporal state, the master-agent activates one or multiple sub-agents of this pyramid. As an innovation, this HRL approach includes deterministic policies that act as sub-controls within the hierarchical structure for sub-tasks that do not require any self-learning mechanics and thus reduce the overall training time. By defining a generalized hierarchical formalism the developed approach is also adaptable to any other hierarchical structure.

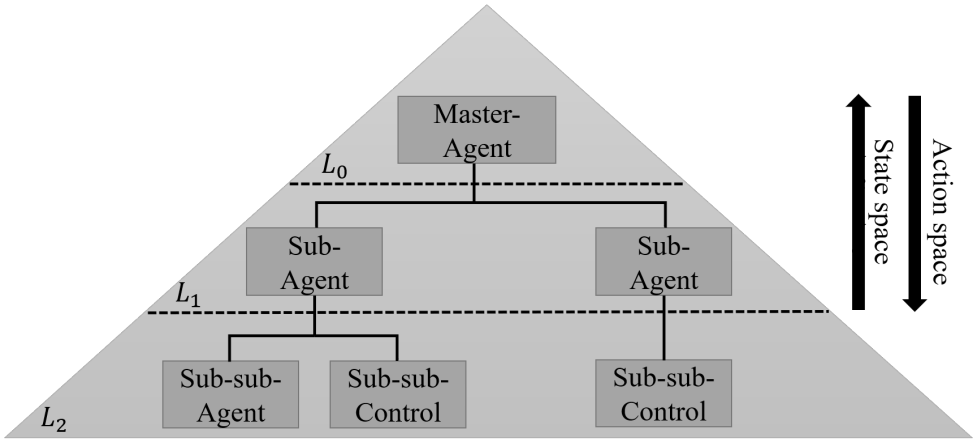

Figure 4.13: Developed HRL framework for e.g. machine control tasks.

The agent on the highest level L_0 chooses a sub-task ω^{L_0}, which is executed by one of the

sub-agents on level L_1. This sub-agent then chooses sub-task $\omega^{L_1} \in \Omega^{L_1}$, which is executed by the sub-sub-agent or one of the sub-sub-controls on level L_2 which finally gives primitive actions to the environment. Sub-tasks ω always refer to sub-goals $g_\omega \in \mathcal{G}_\omega$ and primitive actions are denoted as $a \in \mathcal{A}$. Every agent, except the lowest-level agent on level L_2, iteratively chooses a sub-goal g_ω, processes it through a sequence of actions a until completion and receives a new sub-goal [199]. Unlike other approaches, the developed HRL framework defines the state space and thus the MDP of the different levels individually. As described in Bellmann's "curse of dimensionality" the problem representation grows exponentially in the number of state and action variables [200]. Therefore, the developed approach aims to reduce either the dimensions of the states or actions level-wise. Thus, the master-agent observes the environment extensively and lower levels only partially the environment with a reduced dimensionality where

$$s_t \in \mathbb{R}^{d_{L_0}} > \mathbb{R}^{d_{L_1}} > \mathbb{R}^{d_{L_2}}. \tag{4.6}$$

By that, the number of information about the environment decreases as the hierarchy level descends. However, the composition of the states as well as the level of information of each state element can differ between the levels and also one certain level and is designed to accordingly achieve a certain sub-goal g_ω. In the developed HRL framework the states can either be discrete, continuous, or a combination of both. Contrary to that, the dimension of the level-wise action selection decreases ascendingly from the lowest hierarchical level with

$$a_t \in \mathbb{R}^{d_{L_2}} > \mathbb{R}^{d_{L_1}} > \mathbb{R}^{d_{L_0}}, \tag{4.7}$$

and is also indicated in Figure 4.13. Except for the lowest level, the action space is determined to be discrete and the agent has the choice between different sub-goals respectively sub-agents, or sub-controls. The action space on the lowest level can either be discrete, continuous, or a combination of both. The RL agents on the lowest level are designed as common MDPs and transit to the next state after each time-step, whereas the actions of the higher hierarchical agents persist for an extended period of time. By that, these agents are formalized as a SMDP which allows to temporarily extend actions by denoting the random variable N [200]. The transition probability of a SMDP is thus extended to $P(s_{t+N}|s_t, a_t)$, where N can either be a fixed period of time-steps or is determined by the time that is needed to reach a sub-goal. To allow the entire structure to find an optimal path trough the different levels. The individual rewards of the master agent and sub-agents etc. are manually designed to assists the agents to accomplish the overall task or the individual sub-goals.

Depending on the complexity of the actual task, the decomposition into hierarchical levels and the number of sub-goals can differ. Assuming the structure in Figure 4.13, there is one

master agent on the highest level, two sub-agents on the middle level, where one is connected to one sub-sub-agent and a sub-sub-control and the other one to one sub-sub-control.

Based on the idea of macro-actions $\mathcal{M}$, which are predefined deterministic sequences of primitive actions, sub-controls in the context of HRL are developed. One macro is defined as $m_i = \{a_{i,1}, ..., a_{i,n}\}$ and triggers n total primitive actions [201]. Marco-actions are beneficial for problems, where the agent requires to perform a distinct number of actions in a row repeatedly. Macro-actions can be seen as local policies which are active in certain regions of the state space and thus reducing the entire state space available for agent [202]. The drawback of macro-actions is indeed, that they are additional actions, thus increase the action space and worsens the effect of high dimensionality [201]. Building on that, the developed sub-, or sub-sub-controls etc. embed deterministic polices to allow the agents to not only refer to underlying sub-tasks ω, but also forcing distinct complex behaviors. When the agent in the HRL structure decides to take an underlying sub-control, a deterministic policy Π_{sc} is activated which gives for every state a distinct action as

$$\Pi_{\mathrm{sc}}(s_t) = a \forall s_t \in \mathcal{S}. \tag{4.8}$$

Different to the macro-actions, the sub-control policies are not only based on primitive actions, but rather can be any discrete action or arbitrary continuous functions with respect to single values of the state vector. The length or duration of the activated sub-control can be selected as necessary. The assignment of sub-controls offers benefits for clear and simple task which do not require a certain sub-task with a learning system for accomplishment. It is worth noting that fully trained sub-agents with no further re-training behave like to sub-controls and can then be viewed as deterministic policies.

Unlike other HRL approaches, in this structure the master-agent and sub-agents, etc. are not trained simultaneously to avoid long training times and stability problems. In contrast a simultaneous learning, the training is phased from the bottom to the top $L_2 \rightarrow L_0$. Agents on the same level can be trained individually, but decoupled from each other. Ideally, the sub-agents are trained to an extent that they can henceforth be viewed as sub-controls. Since the framework is developed to deal with real-world or environments with long computation times as occurring with the DEM, the environments of the lower-level agents are simplified and only consist of the relevant components to speed up the computation. Depending on the simplification, as comparable to the previous iterative learning schedule, the sub-agents are pre-trained with simplified environments and later require a re-training while deploying them in the training of the superior agent.

The developed HRL framework is suited for very complex tasks and reduces the complexity by decomposing the task into multiple levels, environments with distinct state and action space and individual sub-goals. By reducing the dimensionality of the state and action space per level,

sufficient training times can be achieved. In this HRL framework, the dimensions of the state space are reduced in ascending order of the hierarchical levels and the action space vice versa. The usage of predestined deterministic sub-control policies reduces the complexity even further and also assists to reduce the overall training time. While training the entire task, a bottom to top training is foreseen, whereas the training yields to convert sub-agents to sub-controls. The number of hierarchical levels, as well as the total amount of sub-agents, sub-controls, etc., is defined individually by the complexity of the desired task and must be specially designed.

4.4 Single-Actuation Transportation Task

The PSM is substantially managed with conventional control architecture and main parts are processed with the PLC or e.g. the positioning of the decentralized drives is made with proprietary controllers. But especially the core functionalities of manipulating parcels and the complex triple interaction between kinematics of the actuator-units, the flexible transport film and the parcels require a data-driven control approach. The proposed controller needs to observe high-dimensional states and give complex actions to successfully manage the manipulation of parcels with peristaltic waves.

With regard to the general operating principle of the PSM in the infeed line of distribution centres, the following mode of actions are supposed to be achieved in the future. Incoming parcels in a bulk constellation are unloaded at the front of the PSM. One actuator-unit singulates distinct parcels, passes them to the subsequent actuator-unit which then transports the parcels to the other end of the transport film, where the parcels are handed over to the attached downstream sortation system. This parallelization of the singulation and transportation process provides the advantage of a fast and gentle operation. Due to the modular basis of the PSM, extensive functionalities with cross transportation and sortation tasks could be achieved as well. This work however focuses on a PSM with one table and two actuator-unit to show the feasibility of the peristaltic principle and the possibilities to solve sophisticated RL tasks with DEM based digital twins.

To show the applicability of the developed methodologies on the one hand and the general usability of DEM for simulating complex machine processes on the other hand, the transportation task of the PSM is presented using two different approaches. Both approaches use the developed RL strategies and recent RL algorithms to achieve reasonable performances. In the first approach, one actuator-unit is controlled to move one parcel along the film. So that, the agent intuitively learns how to move parcels with peristaltic waves. To significantly speed up the training of the RL agent, the distributed ACRL methodology with multiple instances is used. In the second approach, a transportation scenario with one parcel and two actuator-units is described. In a developed HRL framework the two units are controlled to not only move the parcel forward,

but also to hand-over the parcel to the second unit. Therefore, the iterative learning strategy is used to identify the correct DNN structure and learning parameters for the relevant task in both approaches. Additionally, the multi-complex environments of this schedule are used reduce the complexity of the proposed problem and learn the fundamental behavior of the specific task or e.g. learn how to generally move the actuator-unit of PSM within the machine limits.

The elaboration and the results of both approaches are explained in the following and thus underline the particular relevance of coupled DEM-RL approaches. Afterwards, in order to use the developed RL approaches with the real machine, a RL-PLC implementation is discussed.

The single-actuation transportation task as the very fundamental core functionality of the PSM, is required to develop prospective functions and tasks. The challenge of this task is twofold. First, the agent learns to generally move the actuator-unit with its dynamics within the given machine boundaries. Second, the actuator-unit generates a moving wave in combination with the transport film which then manipulates the parcels. As this approach is the very first attempt to move parcels with peristaltic waves there in no variation of the parcel properties or initial parcel position. A successful transportation task is determined by the movement of a parcel from a starting position to the defined end of the transport film. In the following, the developed environments, RL algorithm and results are explained.

4.4.1 Single-Actuation Transportation Environment

The environment of the specific single-actuation transportation is defined by the desired observations and possible actions. Considering the environment as a black-box, it emits a state and can advance to the following state by receiving any action. For the agent, the actual implementation of the environment is not relevant. Since the real PSM cannot be used for the training process sufficiently, a model that similarly behaves to the real application is desired. As previously discussed, the developed digital twin of the PSM modeled with the DEM is used as environment and can properly substitute the real machine. But, before discussing the explicit training, the dynamics of the mechanics of the PSM have to be derived.

Considering the dynamics of the combination of servo drives, gears and linear units of the axes the resulting acceleration and velocity limits are stated in Table 4.1. The change of velocity is therefore given by

$$\Delta v_{\text{slide}} = a_{\text{slide}} \Delta t, \tag{4.9}$$

$$\Delta v_{\text{lift}} = a_{\text{lift}} \Delta t, \tag{4.10}$$

$$\Delta \omega_{\text{tilt}} = \alpha_{\text{tilt}} \Delta t, \tag{4.11}$$

for the translational velocity of the slide v_{slide}, lift axis v_{lift} and the rotary velocity ω_{tilt}, with

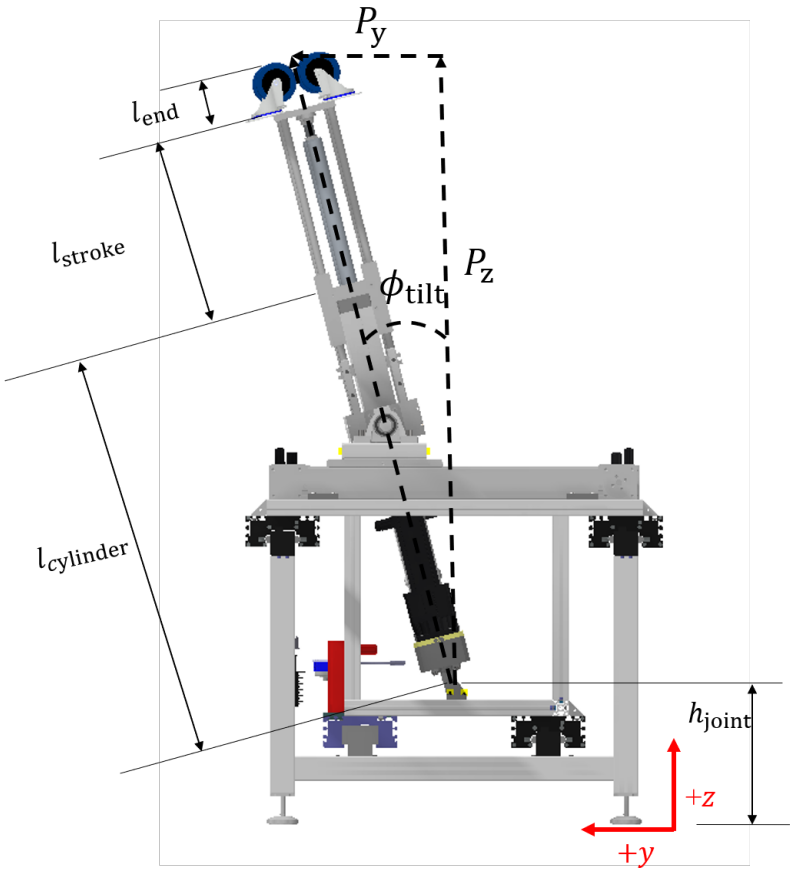

Figure 4.14: PSM mechanics.

their constant accelerations $a_{\text{slide}}, a_{\text{lift}}, \alpha_{\text{tilt}}$ and the considered time step Δt, respectively. Thus, the change of position is equally computed with

$$\Delta s_{\text{slide}} = \frac{1}{2} a_{\text{slide}} \Delta t^2 + v_{0,\text{slide}}, \tag{4.12}$$

$$\Delta s_{\text{lift}} = \frac{1}{2} a_{\text{lift}} \Delta t^2 + v_{0,\text{lift}}, \tag{4.13}$$

$$\Delta \phi_{\text{tilt}} = \frac{1}{2} \alpha_{\text{tilt}} \Delta t^2 + \omega_{0,\text{tilt}}, \tag{4.14}$$

for all of the three axes. These movements of the drives have to be transformed into the mechanics of the PSM. Since the slide axis directly influences the current position of the actuator-unit in X-direction, it follows

$$P_{\text{x}} = P_{\text{x,old}} + \Delta s_{\text{slide}}, \tag{4.15}$$

so that the actual position can be computed. In contrast to that, the positions in Y- and Z-direction depend on the stroke of the lift cylinder and the tilting angle influenced by the

tilt drive. Figure 4.14 shows the mechanics of one actuator-unit, where the total length of the lifting-unit and the angle around the joint influence the actual position in both directions. If the unit is tilted, so that the lift cylinder is no longer parallel to the Z-axis, further tilting or lifting will cause a change of the position in Y- and Z-direction. The entire length of the lifting-unit is given by

$$l = l_{\text{end}} + l_{\text{cylinder}} + l_{\text{cylinder}}, \tag{4.16}$$

with $l_{\text{end}} = 150$ mm, $l_{\text{cylinder}} = 966$ mm and the length of the stroke which is equal to the actual lifting height $l_{\text{stroke}} = \Delta s_{\text{lift}} + s_{\text{lift,old}}$. Therefore, the position in Y-direction is given by

$$P_{\text{y}} = \sin(\phi_{\text{tilt}})l, \tag{4.17}$$

with the tilting angle $\phi_{\text{tilt}} = \Delta\phi_{\text{tilt}} + \phi_{\text{tilt,old}}$. The stroke of the lifting cylinder is constrained to a maximum 550 mm, the tilting angle can vary between $-20° \leq \phi_{\text{tilt}} \leq 20°$. The origin of the PSM coordinate system is defined at the very front of the PSM in X-direction, the middle of the PSM in Y-direction and the ground floor for the Z-direction. Under certain circumstances, the origin in Z-direction is changed to e.g. the height of the film or the lowest possible stroke. Therefore, to determine the position in Z-direction with the mounting height of the joint of $h_{\text{joint}} = 350$ mm, it follows

$$P_{\text{z}} = \cos(\phi_{\text{tilt}})l + h_{\text{joint}}. \tag{4.18}$$

These derived position update equations must be programmed to properly control the actuator-unit in the simulation model. The DEM software tool LS can move certain geometries but does not support special motion packages. In LS it is only possible to move linearly, rotary, or oscillating for a distinct number of time steps. Velocity changes caused by accelerations and positioning to the relative or absolute position are thus not directly implementable in LS. Therefore, the dynamics of the PSM are modeled in Python and the required movements of the end-effector in the DEM are computed by the super-ordinate system. Small deviations caused by the approximation of the acceleration ramps, as explained in Section 4.3.1, are read back to Python to eliminate an integration error.

The described dynamics of the actuator-unit are computed in Python and coupled with the DEM to move the end-effector in the simulation domain. In combination with the discussed individual components as the parcel and the transport film the single-actuator transport environment is set up and shown in Figure 4.15.

As for all developed PSM environments, the initial position of the end-effector and the parcel are freely definable. The length of the transport film is set to $3,500$ mm and the width to

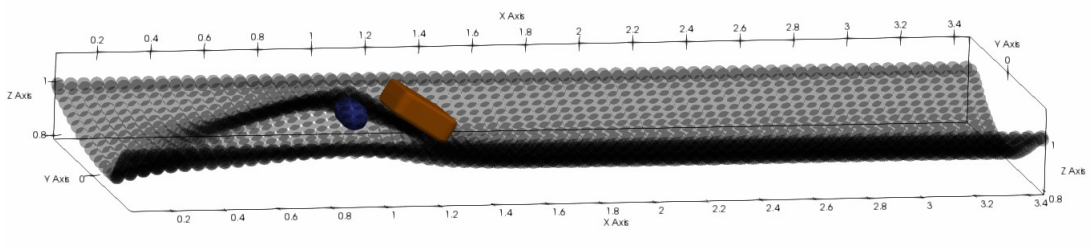

Figure 4.15: Single-actuator transportation environment.

$2,000$ mm which corresponds to the real PSM. Also, the size and weight of the parcel can be freely defined.

4.4.2 Single-Actuation Transportation MDP

The task of the single-actuation transportation is defined as the successful transport of a parcel from a fixed position to the end of the transport film. Therefore, the agent has to create a moving wave which transports the parcel along the film. To allow the agent to successfully solve the task, the state and action space and the reward signal of the MDP are following defined. The state is encoded as the current actuator-unit position $(P_{\mathrm{x}}, P_{\mathrm{y}}, P_{\mathrm{z}})$, the current velocities of the axis of the actuator-unit $(v_{\mathrm{slide}}, v_{\mathrm{lift}}, \omega_{\mathrm{tilt}})$ and the position of the parcel with $(P_{\mathrm{px}}, P_{\mathrm{py}}, P_{\mathrm{pz}})$, concatenated into a single state vector. This definition results in a continuous state space and consequently requires the use of function approximators and suitable RL algorithms. In this task a discrete action space of setting the accelerations of the three drives, namely slide, lift and tilt, is defined. The possible actions for each axis are fully accelerate (1), fully decelerate (-1) and keeping velocity (0) which defines an action set of $3^3 = 27$ possible combinations. The time step for deploying an action is set to 0.2 s, which yields to a dynamic movement of the three axes and the entire actuator-unit.

The initial condition and also the initial state of the environment is defined as the position of the actuator-unit where $P_{\mathrm{x}} = 600$ mm, $P_{\mathrm{y}} = -100$ mm and $P_{\mathrm{z}} = 1,250$ mm. The parcel position is fixed to $P_{\mathrm{px}} = 1,300$ mm, $P_{\mathrm{py}} = 0$ mm and $P_{\mathrm{pz}} = 1,450$ mm and the velocities of all axes are initially set to zero. The parcel has a size of (L x W x H) 200 mm x 400 mm x 100 mm, where the longest edge is parallel to the Y-axis and has a weight of 700 g.

Within the single actuation transportation task, the agent is supposed to learn to move the actuator-unit in the given PSM machine limits. The limits are defined by both ends of the linear unit in X-direction, the maximum lifting height and the maximum tilting angle. Additionally, since the accelerations of the three axes are presented by the agent, an exceeding of the maximum velocities shall be avoided. The transport behavior is learned by giving a positive reward signal for a movement of the parcel compared to the previous time step and also a high reward after successfully finishing the episode. Therefore, an episode is evaluated to be successful when

the parcel is transported into the target area which is defined as the end of the film in positive X-direction. The rewarding is defined by the reward signal of the individual time step with

$$r_t = \begin{cases} 1, & \text{if: } P_{\mathrm{px},t} > P_{\mathrm{px},t-1} \quad (4.19) \\ -10, & \text{if: exceeding machine boundaries} \vee \text{maximum velocities} \quad (4.20) \\ 10, & \text{if: } P_{\mathrm{px},t} > 3,000 \text{ mm} , \quad (4.21) \end{cases}$$

where $P_{\mathrm{px},t}$ is the position of the parcel in X-direction in the current time step. An exceeding of the machine boundaries is therefore penalized with a negative reward. In this task, the environment does not allow the parcel to leave the simulation domain which would be equated to fall off the film and is thus not evaluated. If the boundaries or velocities are exceeded, the episode will be canceled and restarted with the initial conditions. Additionally, the maximum number of steps is set to 200. If the agent reaches this limit the episode will be canceled and the agent will receives the negative reward, too. If the parcel is moved into the target area which means the position of the parcel is $P_{\mathrm{px},t} > 3,000$ mm, the episode will end successfully. This sparse rewarding requires suitable training of the RL agent. The agent has to independently learn how to move the parcel and also obtain a positive transport behavior.

4.4.3 Training of the Single-Actuation Transportation

The used RL algorithm for this task is the A2C algorithm which is explained in detail in Section 3.2.4. To allow the agent to sufficiently learn and also to generalize the desired task, a suitable DNN structure is required. To speed up the training and limit the total number of parameters, the network size should be as small as possible. However, the complex task and high-dimensional state space requires a particular RL training approach. Therefore to reducing the complexity, the explained iterative learning schedule with the multi-complex environments of Section 4.3.2 is deployed. In the iterative learning schedule, the developed DEM simulation represents the complex environment while the simplified environment is based only on the actuator dynamics. The simplified environment, executed in Python, is mainly based on the Equations (4.12, 4.13,4.14) and able to realistically simulate the actuator movement. The transport film and the parcel are not directly simulated in the environment, but the parcel movement is roughly approximated. If the end-effector moves in an area in front of the parcel, described by the parcel position, the position of the parcel is artificially moved forward a little. This movement randomly changes between 55 mm to 250 mm. That is why not only the agent learns to move in the machine boundaries but also to transport a parcel in the forward direction. However, for the agent there is no difference between the simplified and complex environment in the state and action space or concerning the rewarding.

By alternating between the simplified and complex DEM environment with the iterative learning schedule, an optimal DNN structure and learning parameters of the agent are determined.

The training starts in the simplified environment until the training is successful and the task is accomplished. If after $5,000$ episodes the training is not successful, the DNN structure will be increased in depth and width with the described update rules. After a successful training, the agent is re-trained with the complex DEM environment for 150 episodes. If the agent can not achieve successful runs with it, the DNN structure is increased again and restarted with the simplified environment. By performing this iterative learning schedule, starting with one layer and 5 neurons, the following DNN structure for the actor and critic path yield to sufficient results. Both, actor and critic, consist of two hidden layers with 30 and 35 neurons, respectively and the ReLu activation function. The output layer of the actor is set to the softmax activation function to define the discrete actions space. The critic output layer is set to the linear activation function. Additionally, to increase the sample efficiency, ER with a replay buffer of size $n = 100$ is applied. According to Section 4.3.3, the replay buffer is filled with experience of previous state-action transitions using the FIFO principle. In the used ER, a batch of 50 randomly chosen samples out of these 100 is used to train the agent accordingly. The discount factor is set to $\gamma_{\mathrm{d}} = 0.99$ and the learning rate is set to $\alpha_{\mathrm{lr}} = 5\mathrm{e}{-5}$ for the actor and critic network.

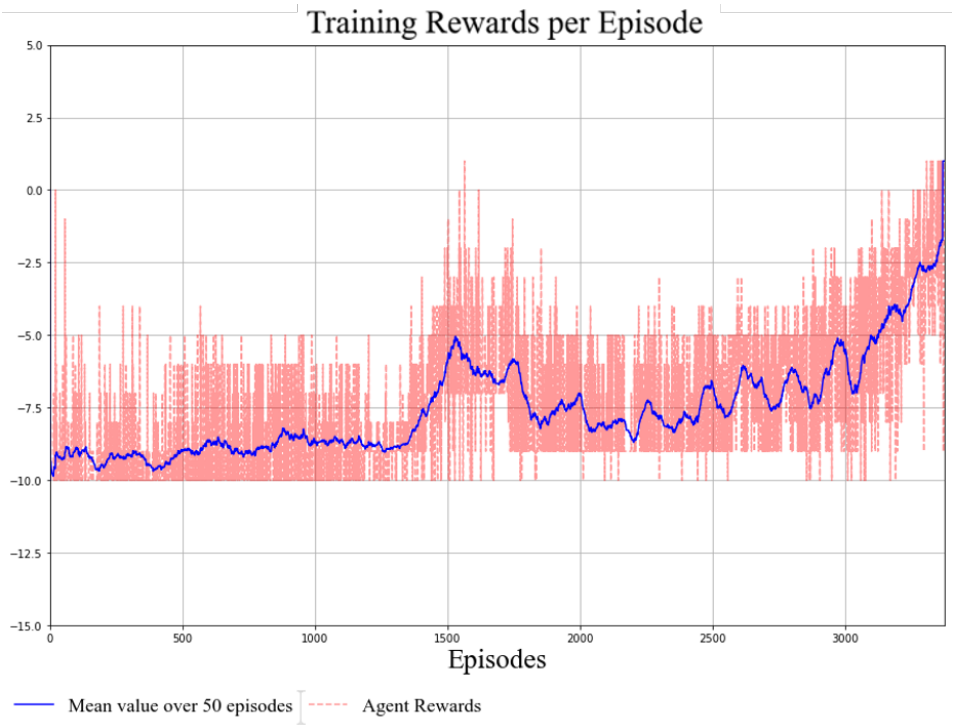

Figure 4.16: Result of the training with the simplified environment.

The final results of the training with the simplified environment are shown in Figure 4.16. During the iterative learning schedule and alternating between both environments, the DNN size has been accordingly adjusted. Over within the training duration of $3,375$ episodes the agent gradually increases the cumulative rewards and successfully transports the parcel in the simplified environment. As shown, the simulated parcel behavior is noticeable in the stairs-like

course of the average reward. Every time the agent moved the actuator-unit in front of the parcel, it is pushed forwards and the agent received a positive reward. This rewarding helps the agent to fundamentally learn how to transport a parcel. As foreseen in the iterative learning schedule, the episode length is fixed, but a successful training episode stops the training and alternates to the next step. As anticipated in the schedule, the agent is now shifted to interact with the DEM environment. Considering the environments as black-boxes, the agent is not able to distinguish between the multi-complex environments and interacts using the same state and action space.

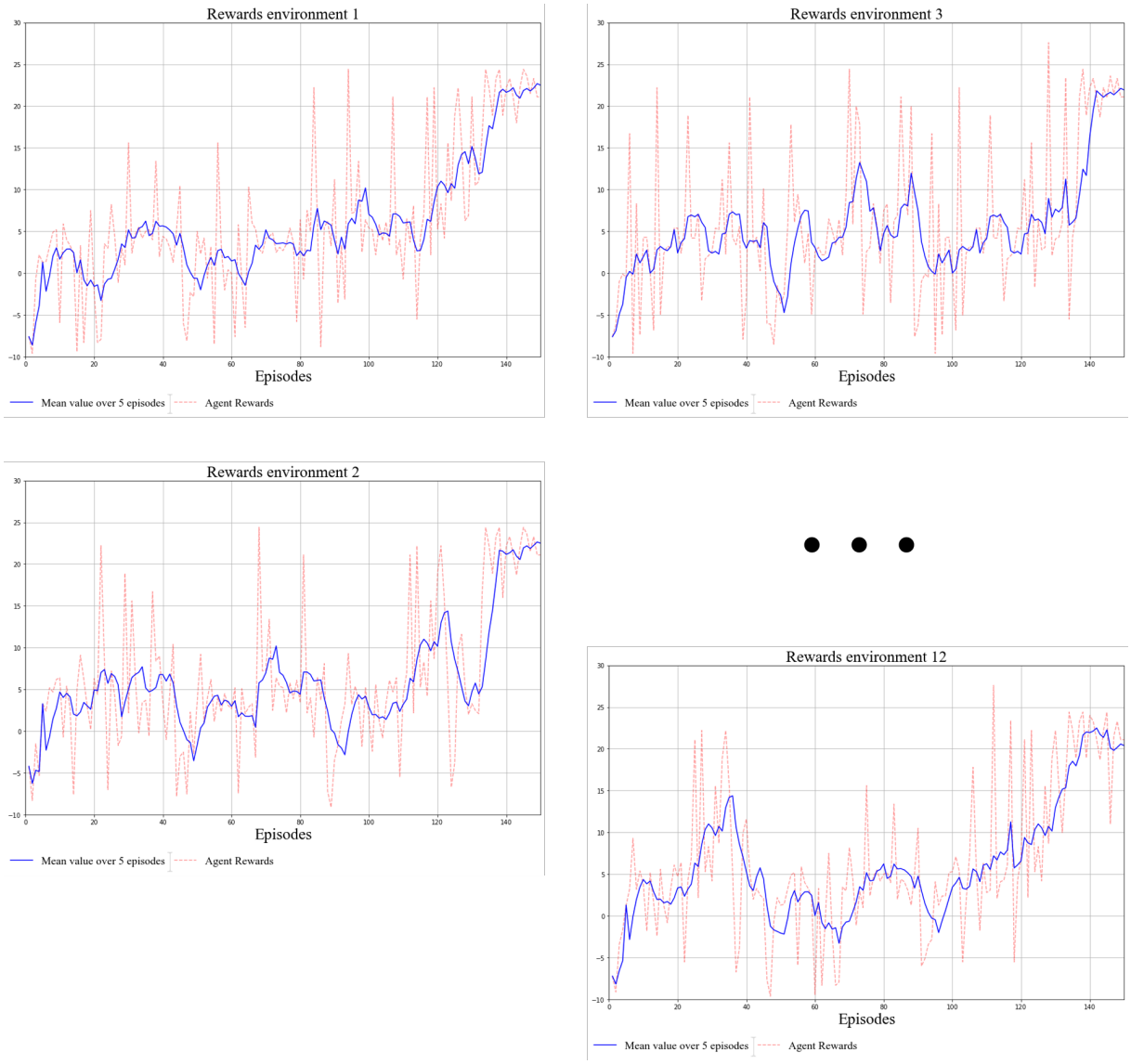

Figure 4.17: Result of the distributed ACRL approach with 12 PSM-DEM environments.

In this second phase of the iterative learning schedule, the distributed ACRL, described in Section 4.3.3, is used to properly re-train the agent. There, $k = 12$ individual environments, each consisting of the described DEM environment, asynchronously interact with one ACRL agent. The replay buffer is thus accordingly filled with the experience of all of these environments.

The advantages of applying the distributed ACRL are twofold. First, the number of samples using multiple environments is increased significantly, which is of particular importance in the exploring phase at the beginning of the re-training. Second, in combination with ER, the training is stabilized and the performance of sample efficiency is increased which leads to faster convergence. The agent is trained over 150 episodes for each environment which results in $1,800$ episodes of re-training. The cumulative rewards of certain environments are depicted in Figure 4.17. It is shown that the re-training benefits from the pre-training because there are barely episodes that end with the penalty of -10. It is assumed that the agent previously learned to control the actuator-unit in its boundaries, which is indeed a complex task considering that the agent has to learn to stop the unit before it exceeds any boundaries with the resulting mechanics and constraints. Besides, the general trend in all the subplots of the environments indicates a similar course. Although the individual environments are at different stages within the episode, the agent can decide for sufficient actions. Finally, the last $10 - 15$ episodes are all successful runs which correspond to a converged learning process.

4.4.4 Summary

The described approach of solving the single-actuation transportation problem with RL provides the basic work of a successful RL-DEM coupling. The developed digital twin of the PSM consisting of the three main parts - the mechanical design, transport film and parcel realistically simulates the PSM behavior. This digital twin is successfully coupled with the RL framework and the DEM entirely controlled via Python.

The single-actuation transport task is tackled using the iterative learning schedule with a simplified environment, mainly modeled with the actuator dynamics, and with the complex DEM environment facing all details of this complex task. The actuator-unit is controlled by changing the accelerations of the three axes to properly move in the machine boundaries and transport the parcel to the target area, which is located at the end of the film. By using the iterative learning schedule the size of the DNN structure and the learning parameters are defined. Additionally, the iterative learning schedule reduces the complexity of the sophisticated and high-dimensional control task and is particularly beneficial in the exploring phase of the RL training. The deployed distributed ACRL approach together with ER yield to promising results and fast convergence. The chosen rewarding allows the agent to properly learn the transportation of parcels with peristaltic waves and thus solve the desired task. Manual tests with the environment have determined $5 - 8$ s for comparable fast transportation of the parcel in this configuration. The agent has achieved to control the actuator-unit to transport the parcel in the target area in 7.3 s. The entire computation of the iterative learning schedule and the re-training with the 150 episodes per environment took 72 h in sum. Assuming the agent is trained from scratch with the same number of episodes, but without pre-training with the simplified environment, it would

approximately take 30 days. Considering that in this period of time the DNN size of the agent has to be tuned and the environment tested, the training time would be further extended and this would yield an unfeasible training situation.

4.5 Multi-Actuation Transportation Task

The second application of the machine control of the PSM not only focuses on the transport of a parcel with multiple actuators, but also provides the basis of incorporating future tasks of the PSM. The challenges of this transportation task are the interaction between two actuator-units, avoidance of possible collisions and the orchestration of the two units. For further purposes the developed control approach is supposed to be extendable to n_{act} units. Furthermore, a generalized behavior of transporting a parcel in a random parcel position is developed. Similar to the previous chapter, a digital twin of the PSM is used to train the RL agent. However, incorporating multiple actuator-units again increases the state space significantly which requires the deployment of a hierarchical RL structure as explained in the Section 4.3.4. Additionally, the iterative learning schedule with multi-complex environments to train individual RL agents is used to solve this very complex task.

4.5.1 Multi-Actuation Transportation Environment

To outline the multi-actuation transport environment plus the relevant state information and actions, the explicit task and the general structure of the HRL framework are described first. In the environment two actuator-units initially start at both ends of the transport film. One parcel is randomly placed on this film. As learned from the previous approach and also described in [184], a parcel movement is generated by lifting up the unit, creating a wave in front of the parcel which then moves it forwards. This means, a parcel transportation is given when the actuator-units are in front of the parcel and meet a corresponding lifting height.

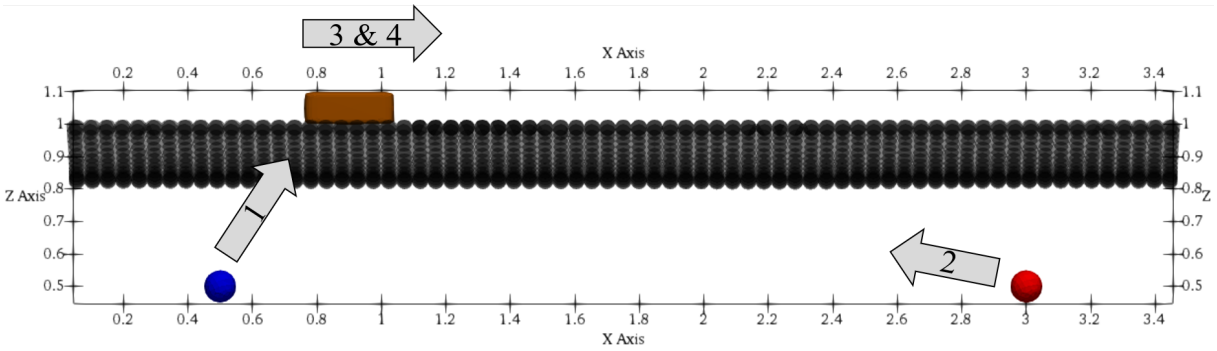

Figure 4.18: Multi-actuation transportation task.

The whole multi-actuator transport environment is shown in Figure 4.18, with the first (blue) and the second actuator-unit (red). In the initial condition the positions of the actuator-units and the parcel are determined and a master-agent tries to find the fastest strategy to move the parcel along the film. Hence, the position of the parcel plus the maximum allowed velocities of the actuator-units are of importance. In the developed HRL framework one master-agent is responsible for activating one of the units which then moves to the transportation position accordingly and transports the parcel along the film. The master agent learns to decide to use only the second actuator-unit or a parcel exchange between both units to transport the parcel as fast as possible along the film. Therefore, the entire multi-actuation transposition problem is divided into four sub-tasks which can be activated by the master-agent. In the sub-tasks one and two the explicit actuator-units are forced to reach suitable transport positions in front of the parcel. The movement of the activated actuator-unit is controlled by one specific sub-agent, whereas the other is controlled by another sub-agent. If one of the units reach the transport position, the master-agent will decide to actively transport the parcel forwards, which is implemented by a hard-coded moving of the unit in positive X-direction for both of the units. The forward movements are assigned to the sub-task three for the first actuator-unit and the fourth for the second, respectively. These forward movements then transport the parcel along the film for a certain time-step. Depending on the current state with the actual position of the parcel, the fastest transportation can be achieved by only using the second actuator-unit. In other cases it is advantageous if the parcel is handed over from the first to the second actuator-unit which then transports the parcel to the end of the film.

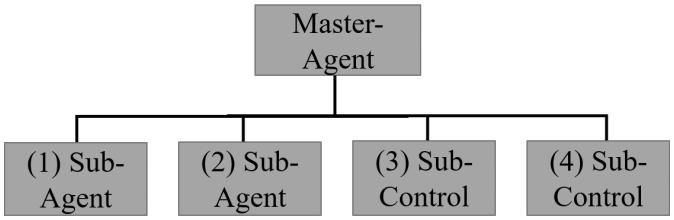

Figure 4.19: HRL framework for the multi-actuation transportation task.

Based on the described multi-actuation transportation problem and the four sub-tasks, the developed HRL framework is composed of one master-agent, two sub-agents and two sub-controls. Therefore the set of sub-task policies is defined as

$$\Omega = \{\omega_1, \omega_2, \Pi_{\mathrm{sc}_1}, \Pi_{\mathrm{sc}_2}\}, \tag{4.22}$$

with ω_1, ω_2 as the learned policies of the sub-agent and $\Pi_{\mathrm{sc}_1}, \Pi_{\mathrm{sc}_2}$ as the deterministic functions of the sub-controls.

The framework shown in Figure 4.19 has two levels with individual MDPs and independent state and action spaces. Since the HRL framework of this task is designed to control n_{act} actuator-units of the PSM, parallel tasks might be necessary, too. But, to limit the complexity and show the general applicability of the developed approach, only one sub-task referred to one actuator-unit is always active. The master- and sub-agents can be trained separately and are interchangeable when the problem description changes. To speed up the training with the DEM environment, the sub-agents are purely, and the master-agent is pre-trained with a simplified Python environment. To achieve an appropriate performance with the DEM environment and the complex interaction of the units, transport film and the parcel, the master-agent is re-trained. As described in the developed novel HRL framework, sub-agents which are completely trained are deployed as deterministic functions. This further reduces the total number of iterations and computation time along with the re-training phase of the master-agent which incorporates the complex DEM environment.

The explanation of the entire multi-actuation transport environments follows the subdivision into the two layers of the HRL framework, starting with the environment and MDPs of the first and second sub-agent and finally the master agent. Additionally, the implementation and definition of the sub-controls is explained as well.

Environment of Sub-Task 1 and 2

The sub-goals of both sub-tasks $g_{\omega_1}, g_{\omega_2}$ are to move the first respectively the second actuator-unit in front of the parcel. The kinematics of the environments for both sub-tasks are equal to the previous single actuation transportation task. Both environments are called as two instances with individual pre-parameters and communicate over the standardized interface to the coupled RL sub-agents. Therefore, the continuous state space for both tasks is defined as the position of the actuator-unit and the parcel position But, to directly force the agent to control the unit in front of the parcel, the target position in X-direction is set to at least $P_{\mathrm{tx}} = P_{\mathrm{px}} - 300$ mm in front of the actual parcel position. The parcel position is randomized in the ranges of

$$\begin{aligned} -800\ \mathrm{mm} < & P_{\mathrm{px}} < 3,000\ \mathrm{mm}, \\ -500\ \mathrm{mm} < & P_{\mathrm{py}} < 500\ \mathrm{mm}, \\ 1,000\ \mathrm{mm} < & P_{\mathrm{pz}} < 1,300\ \mathrm{mm}, \end{aligned} \tag{4.23}$$

to allow the agent to reach every parcel located at any place on the film. Therefore, it is pre-checked if the end-effector of the actuator-unit is able to reach the target position considering the given constraints. If it is not possible to reach the position, another random parcel position will be computed. The initial position of the first actuator-unit, more precisely the end-effector of it, is set to $P_{\mathrm{x},1} = 500$ mm, $P_{\mathrm{y},1} = 0$ mm and $P_{\mathrm{z},1} = 500$ mm. The initial position of the second actuator-unit is set to $P_{\mathrm{x},2} = 3,000$ mm, $P_{\mathrm{y},2} = 0$ mm and $P_{\mathrm{z},2} = 500$ mm.

MDP of Sub-Task 1 and 2

The concatenated continuous state vector consists of six elements representing the position of the individual actuator-unit and the desired target parcel position. Instead of to control the velocity of the three axes of the actuator-unit, which in indeed a highly complex task, while predicting an exceeding of the machine boundaries etc., in this task the actions are defined as a relative positioning in the three-dimensional space. In the relative positioning the desired position of the actuator-unit is computed with the Equations (4.12, 4.13, 4.14). The axes are positioned with a total time step of 0.5 s. During this time step the individual servo drives accelerate and decelerate for 0.25 s to reach the new position and always end up with zero velocity for all the axes. To allow a continuous relative movement, the acceleration values of the drives are proportionally set by the primitive actions. The continuous action space is therefore defined by a vector of three elements $a_{\mathrm{x}}, a_{\omega}, a_{\mathrm{z}} \in [-1, 1]$ for the slide, tilt and lift drive, and is then accordingly transformed to meaningful acceleration values in the environment. By this transformation and incorporation of the accelerations limits of the three drives, the dynamics of the actuator-unit and the time step yield a maximum range of movement per axis in

$$\begin{aligned} -62.5 \text{ mm} < &\Delta s_{\text{slide},1} < 62.5 \text{ mm}, \\ -0.03125 \text{ rad} < &\Delta s_{\text{lift},1} < 0.03125 \text{ rad}, \\ -9.375 \text{ mm} < &\Delta \phi_{\text{tilt},1} < 9.375 \text{ mm}, \end{aligned} \tag{4.24}$$

for the first actuator. To increase the difficulty of the multi-actuation transportation task, the range of the relative positioning of the second actuator-unit is increased by changing the pre-parameter set. This means that both actuator-units can move different distances in a certain time which is recognized by the master-agent. The range of the relative positioning of the second actuator-unit is set to

$$\begin{aligned} -125 \text{ mm} < &\Delta s_{\text{slide},2} < 125 \text{ mm}, \\ -0.05625 \text{ rad} < &\Delta s_{\text{lift},2} < 0.05625 \text{ rad}, \\ -18.75 \text{ mm} < &\Delta \phi_{\text{tilt},2} < 18.75 \text{ mm}. \end{aligned} \tag{4.25}$$

As a result the actuator-units are controlled by the sub-agents to move as fast as possible in front of the position of the parcel. If the target position is reached, the master-agent will then switch to a sub-control and transport the parcel forwards. That is why, a rewarding for the proper reaching of the target position has to be defined. The reward signal is mainly influenced by the distance between the positions of the actuator-unit and the target, measured in the three directions and rewarded with

$$r_{\mathrm{dis},t}\begin{cases}1, & \text{if: actuator moves away from the target along all 3 axes} \quad (4.26)\\ 2, & \text{if: actuator moves towards target along at least 1 axis} \quad (4.27)\\ 3, & \text{if: actuator moves towards target along at least 2 axes} \quad (4.28)\\ 4, & \text{if: actuator moves towards target along at least 3 axes,} \quad (4.29)\end{cases}$$

with e.g. the evaluation in X-direction computed with $|P_{\mathrm{x},t} - P_{\mathrm{px},t}| < |P_{\mathrm{x},t-1} - P_{\mathrm{px},t-1}|$. Additionally, a flag d_{limits} evaluates a possible exceeding of the machine limits. If the relevant actuator-unit moves in the valid area and no exceeding is detected this flag will be set to $d_{\mathrm{limits}} = 0$. In sum the final reward signal is composed to

$$r_t = \begin{cases}-3 + r_{\mathrm{dis},t}, & \text{if: } d_{\mathrm{limits}} = 0 \quad (4.30)\\ -10, & \text{if: } d_{\mathrm{limits}} = 1 \quad (4.31)\\ 100, & \text{if: } d_{\mathrm{done}} = 1. \quad (4.32)\end{cases}$$

If the actuator-unit does not move closer to the parcel in all three axis, it will receive a small negative reward, which hence forces the agent to take the shortest way to the target. The done-flag indicates the successful ending of an episode and is thus the final goal of this task. The done-flag will be set to one, if the actuator-unit is moved into a buffer zone in front of the parcel. In contrast to a crisp target value, the buffer zone facilitates the training process. The buffer is defined as cubic zone around the target parcel position with an edge length of 30 mm in X- and Z-direction and an edge length of 10 mm in Y-direction.

Training of the Sub-Agents

Before discussing the training of the sub-agents, the used RL algorithm is explained. For solving the sub-tasks and moving the actuator-units in the right position in front of the parcel, a PPO RL algorithm is used. As already discussed in Section 3.2.4, PPO performs well compared to state-of-the-art RL algorithms and is easy to tune and implement, too. PPO is based on the actor-critic architecture consisting of two individual DNNs for the actor and the critic part, as many other model-free RL algorithms. Knowing that there are also approaches of multi-head RL algorithms with partly shared networks for actor and critic as in [203], separate networks are discussed in this work. The actor-network is build of three hidden layers each having $512, 1024, 512$ neurons and ReLU activation functions in between the layers. The output activation function is set to hyperbolic tangent (tanh) to limit the actions in the range of $[-1, 1]$. The critic network is made of two hidden layers each with 256 neurons, ReLu activation functions between the layers and a linear output layer. The selection of the number of layers, neurons and intermediate layer activation function is a crucial task and strongly affects the performance of the developed RL agent [204]. In an empirical process, the discussed DNN structure yields the best performances

in the test series. The hyperparameters for both networks are empirically determined and set to the discount factor of $\gamma_{\mathrm{d}} = 0.9$ and the learning rate of $\alpha_{\mathrm{lr}} = 1\mathrm{e}{-5}$. The clipping factor is set to $\epsilon_{\mathrm{clip}} = 0.2$, the number of epochs of training the model with sampled data is set $K_{\mathrm{epoch}} = 7$ epochs and the model update is instigated after every 500 time steps.

The sub-agents for both tasks are individually trained since they start from different initial positions and reach different maximum relative movements per step. Since the agent aims to move to the parcel position and no interaction between the mechanics, film and parcel is necessary, the training is completely conducted with the described Python environment. In each episode, the parcel position is randomly selected and after each step slightly changed while adding a small noise signal. This noise is supposed to increase the robustness of the agent, especially later in the DEM environment considering a moving and flexible transport film. The maximum number of steps is set to $1,000$ to ensure a proper exploration of the entire state space and finally reaching the target position in front of the parcel. The agents are trained for $15,000$ episodes, but as shown in Figure 4.20, the first sub-agent converged after $\approx 5,000$ episodes. The reward course gradually increases after a short exploration phase collecting negative rewards.

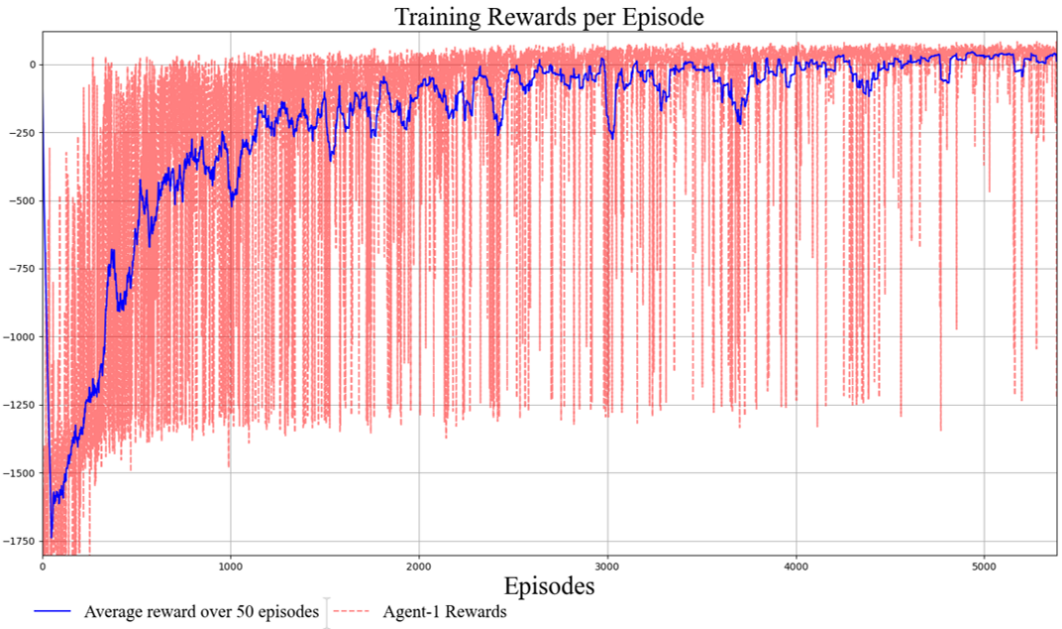

Figure 4.20: Training results of the first sub-agent.

The training of the second sub-agent differs in the initial position of the second actuator-unit and in the optimal episode which is larger compared to the first one. The second actuator-unit has to move beneath and along the position of the parcel to reach the desired target position. This behavior also appears in the training of the second sub-agent as depicted in Figure 4.21 with the first 500 episodes where nearly no positive episode could be achieved. Within the total duration of $15,000$ episodes the training converges after $\approx 3,000$ episodes. The training results and the

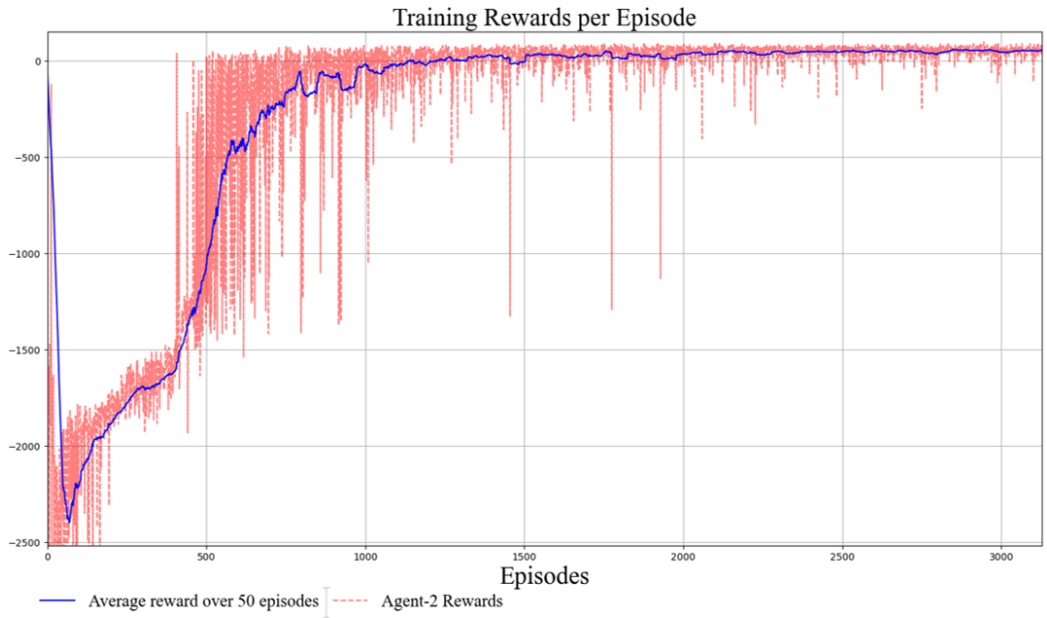

Figure 4.21: Training results of the second sub-agent.

trained agents respectively, are saved and later deployed in the re-training of the master-agent with the DEM environment.

From the prospective of the master-agent, the fully trained sub-agents and sub-controls are assumed as deterministic functions. Therefore, the step-wise training of the different levels of the hierarchical pyramid is beneficial speeds up the training drastically.

Sub-Controls

Assuming, that the sub-agents move the actuator-units to an ideal transport position in front of the parcel, the purpose of the sub-controls is to successfully transport the parcel forwards along the film. In certain cases it might not be necessary to define a subsequent sub-task with a sub-goal and a deterministic action is sufficient. Since the sub-control requires no additional training and sub-controls reduce the available state space for the agent, the usage of sub-controls is a means to reduce the entire training time.

Both sub-controls $\Pi_{\mathrm{sc}_1}, \Pi_{\mathrm{sc}_2}$ can be chosen by the master-agent in the HRL framework and cause a distinct movement of the first and second actuator-unit. With these sub-controls the master-agent is not only able to chose a subsequent sub-task, but immediately force a distinct behavior acting to the environment. Here, $\Pi_{\mathrm{sc}_1}(s_t) = a_{\mathrm{act1}} \forall s_t \in \mathcal{S}$ and $\Pi_{\mathrm{sc}_2}(s_t) = a_{\mathrm{act2}} \forall s_t \in \mathcal{S}$ results always to the fixed actions a_{act1} or a_{act2}, respectively.

In this implementation a_{act1} or a_{act2} are defined as a positive movement of the certain actuator-units in X-direction. Thus, the relevant unit is moved based on the defined limits in the Equations (4.24, 4.25), in positive X-direction for either 62.5 mm or 125 mm in one time-step and reaches a new position.

In future works, the simple sub-control can be exchanged by more sophisticated algorithms like a transport control which observes the parcel and controls the desired trajectory.

Environment of Master-Agent

In Figure 4.22 the entire DEM environment of the multi-actuation transportation task is shown. In general, numerous actuator-units in row, or parallel on other tables are implementable and simulateable considering the modular design of the PSM. However, if more than one actuator-unit manipulates the transport film, the real behavior of the film drastically changes compared to the single-actuator transportation. The wave created by one unit is not locally bounded but extends over the entire film and influences the waves of possible other actuator-units. Hence, the prediction of a successful parcel transport is more complex. Therefore, generating a simplified environment with a good approximation of this behavior is only possible to a certain extend while the DEM simulation is fully capable of simulating the behavior of the film and the parcels.

The main task Γ to be accomplished followed be the policy $\pi_\Gamma \hat{=} \pi_{\mathrm{HRL}}$ is defined as the parcel transport over the length of the film incorporating at least one actuator-unit. Depending on the current position of the parcel, the master agent hat to orchestrate the actuator-units to find the fastest route to move the parcel along the film. Since the DEM environment consisting of the two actuator-units, the flexible film and the parcel is very complex and computationally slow, the iterative learning schedule with a simplified environment is used as well, to reduce the complexity of the desired task and decrease the training time.

Since the dynamics of the actuator-units are already available in Python due to the previous training of the sub-agents, the environments of the sub-agents can be merged and adapted. Only the parcel behavior needs to be extra approximated. It is assumed, that if one of the actuator-units is in the buffer zone in front of the parcel and the master-agent decides to take the allocated sub-control of this unit, the parcel will be moved in positive X-direction. The parcel is transported in the same range of the actuator-unit being moved forwards. Considering a perfectly aligned parcel and a suitable lifting height of the actuator-unit, these assumptions approximate a positive transport. Nevertheless, the agent needs to be properly re-trained in the DEM simulation to achieve a multi-actuator parcel transport with the complex environment.

MDP of Master-Agent

The master-agent, as the overall supervisor in HRL framework of the multi-actuation transport task, has to observe the complete task and decide about the activation of the sub-agents or sub-controls. Therefore, the continuous state transmitted to the master-agent consists of the position of the first and second actuator and the parcel position, which yields a state vector of nine elements. The action space is defined to be discrete and individually activates the four sub-tasks based on the Equation 4.22 namely the sub-agent actuator-unit one, sub-agent

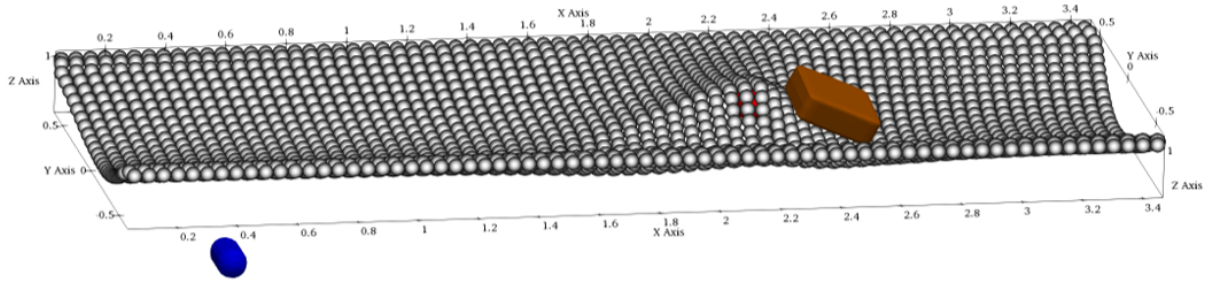

Figure 4.22: DEM environment of the multi-actuation transportation task.

actuator-unit two, sub-control transport actuator-unit one and sub-control transport actuator-unit two. The activated sub-tasks are then performed for a time step of 0.5 s and the state advanced accordingly. Since the sub-agents and sub-controls only partially observe the environment and have no information about the other actuator-unit, the master-agent has to orchestrate the units and ensure a collision-free transportation.

To allow the master-agent to learn an appropriate behavior with the multi-actuator transport environment and successfully solve the task, a proper rewarding structure is necessary. The reward signal is based on the approach of the parcel transportation position, the parcel movement and avoiding collisions between the actuator-units with the following structure

$$r_t = \begin{cases} -10, & \text{if: } |P_{\text{x1,t}} - P_{\text{x2,t}}| < 500 \text{ mm} \quad (4.33) \\ 1, & \text{if: } P_{\text{px},t} > P_{\text{px},t-1} \quad (4.34) \\ 100, & \text{if: } P_{\text{px},t} > 3,000 \text{ mm}. \quad (4.35) \end{cases}$$

The agent receives a negative reward when the actuator-units in X-direction are too close to each other gets a positive reward when the parcel is moved forward compared to the previous time step and a very high reward when transporting the parcel to the end of the film. Additionally, in order to provide the fastest parcel transport with both units, the agent receives a negative reward of -2 for actions which do not yield the aforementioned rewards.

4.5.2 Training of the Multi-Actuation Transportation

The training of the master-agent of the multi-actuation transportation task is applied using the iterative learning schedule in which the training with the simplified environment is used to pre-train the agent and identify suitable DNN structure of the used RL algorithm. Besides a PPO algorithm with a separate network for actor and critic path is selected for the master-agent. As in output of iterative learning schedule the final actor-network is built of three hidden layers each having $512, 1024, 1024$ neurons and ReLU activation functions in between the layers. The output activation function is set to softmax for choosing the action of the highest probability. The critic network is made of two hidden layers with 128 neurons, ReLu activation functions

between the layers and a linear output layer. These configurations achieved promising results in the pre-training and also later in the re-training procedure. The learning rate of the PPO master-agent is set to $\alpha_{\text{lr}} = 1\text{e}{-5}$ and the clipping factor to $\epsilon_{\text{clip}} = 0.2$. The network update is applied every 500 time steps, for $K_{\text{epoch}} = 7$ epochs and with the sampled data in a batch of the size of $n_{\text{batch}} = 64$ steps.

In cases where the first actuator-unit picked up the parcel and the master agent then decided to hand over the parcel to the second one, both units were close to a collision and the episodes were canceled. In RL, and in particular with new developed environments, the chosen rewarding structure or environmental peculiarities can prevent a successful training. To avoid collisions in this scenario and enable a parcel transfer between the units, a special functionality is implemented in the environment. When the parcel is transported by the first actuator-unit and the parcel position becomes $P_{\text{px}} > 1.5$ m, a reverse function is activated. When the parcel passes this limit, the first actuator-unit is moved back for 0.5 m in X-direction and allows the master-agent to use the second actuator-unit to overtake the parcel and transport it to the end of the film.

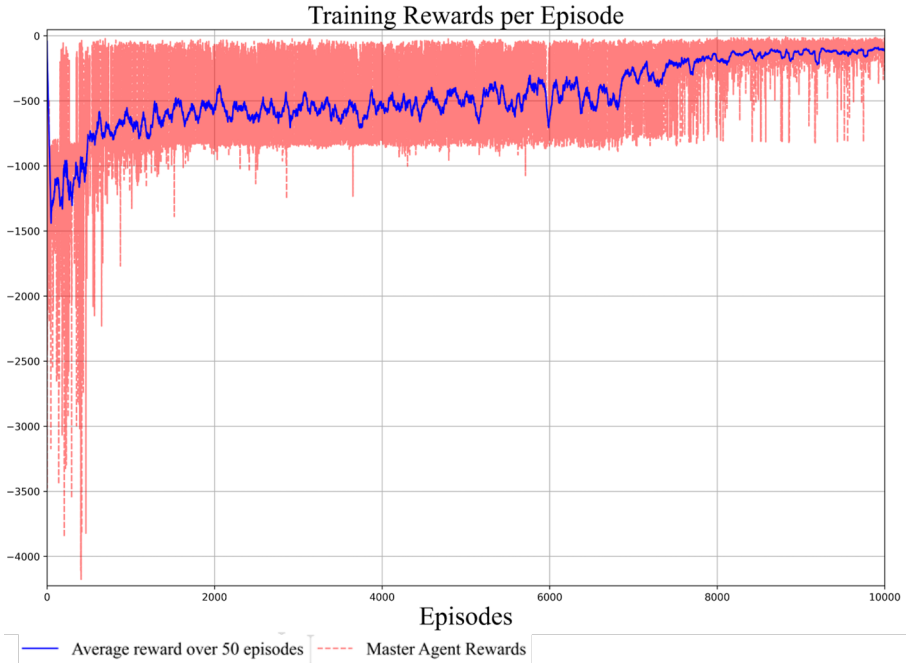

Figure 4.23: Pre-training results of the master-agent.

While running the training multiple times with a total number of 500 time steps per episode and a maximum of $10,000$ episodes, all hyperparameters are alternated to yield promising results. The final reward course of the pre-training of the master-agent is shown in Figure 4.23 and starts with very high negative rewards. In order to increase the exploration especially in the beginning, the episodes up to $1,000$ are only terminated when the maximum number of steps is reached. Therefore, collisions of the units cause big negative rewards, but the agent can continue and explore the state space in the beginning. Subsequently the reward course gradually increases and

reaches satisfying performance in the last $2,000$ episodes. The master-agent learned to deal with the reverse function and transfer the parcel between both units most of the times.

There is a chance that the performance could be increased with further pre-training, but since the simplified environment is only an approximation, the re-training with the DEM environment is still mandatory. Due to the high-dimensional state space a the optimal strategy, indicating the best solution for all cases cannot be determined. Therefore, the agent is tested for 30 episodes with the complex DEM environment after the pre-training and later compared to the re-trained agent. Only a small number of test episodes are carried out because the computation time with the DEM of the multi-actuation transport environment very long and shall only show the current performance of the training progress. In each episode the parcel position is randomly set.

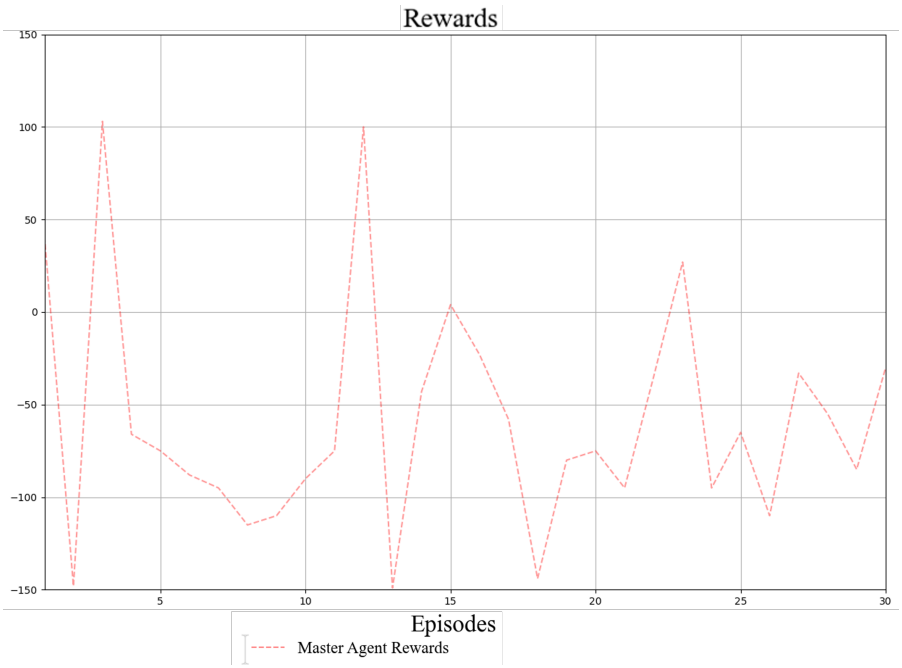

Figure 4.24: Baseline results of the pre-trained PPO master-agent performed with the DEM environment.

After the pre-training with the simplified environment and achieving sufficient performance, the agent is deployed in the DEM environment. In the simplified environment no interactions of the flexible transport film or the parcel behavior are simulated. It only roughly approximates the parcel movement in one direction, but allows the master-agent to handle multiple units and decide between the sub-agents and sub-controls. Thus, it is not assumed that the agent copes well with the DEM environment, nevertheless it might still be able to transport the parcel sufficiently. To test the performance of the pre-trained agent with the complex DEM environment, the agent is deployed to transport 30 randomly set parcels. The reward course of this baseline test is shown in Figure 4.24. Since the reward is mainly connected to the successful parcel transport, which is defined by a positive change of the parcel position in X-direction per time step, the bias of

the rewards with the DEM environment is higher compared to the pre-training. The simplified environment only contains of a finite number of positive discrete transport steps while with the DEM simulation a continuous parcel movement is rewarded. However, the baseline results show that the parcel could only be transported sufficiently two times. In all other cases the episode was terminated due to a collision of the units or reaching the maximum number of steps. To increase the performance of the master-agent, it is re-trained with the DEM environment.

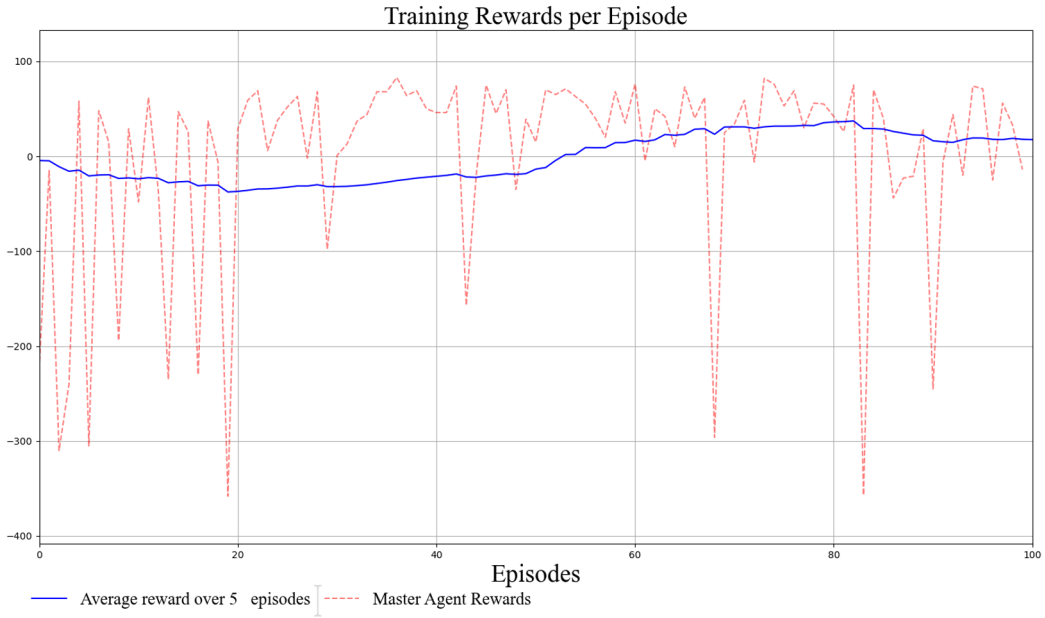

Figure 4.25: Results of the re-training of the master-agent with the DEM environment.

Since the average time for a single episode with the Intel Xeon CPU E5-2697 with 32 cores and 64 GB RAM takes approximated 3 h, the re-training includes 100 episodes to limit the required computation time. In future work the distributed ACRL method can be applied to speed up the training considerably. The training results are depicted in Figure 4.25 and clearly show a learning behavior during this small number of training epochs. The master-agent starts with considerably small rewards and increases them over the number of episodes. Besides, it is shown that the number of failed episodes appear more frequently in the first quarter of the entire training. At the end of the re-training the master-agent successfully finished the multi-actuation transport task most of the times.

The re-trained agent is subsequently deployed again with the DEM-environment for 30 episodes for comparison with the baseline results. In Figure 4.26 the reward course after the re-training is shown, which clearly consists of higher rewards compared to the baseline. The master-agent achieved a successful transportation 25 times. Since the parcel is randomly set in every episode, the individual episodes between the baseline and re-trained results are not comparable, but the overall trend shows that the re-training significantly increased the performance of the master-agent. However, the agent could not successfully transport the parcel with both actuator-units

three times, which could be improved by a further re-training with more episodes. Also, it is important to note, that the sub-agents, which manage the low-level control of the actuator-units are not re-trained with the DEM simulation and are deployed as deterministic functions based on their gained knowledge with the simplified environments. Poor actions in the low-level control also lead to insufficient performance on the higher level.

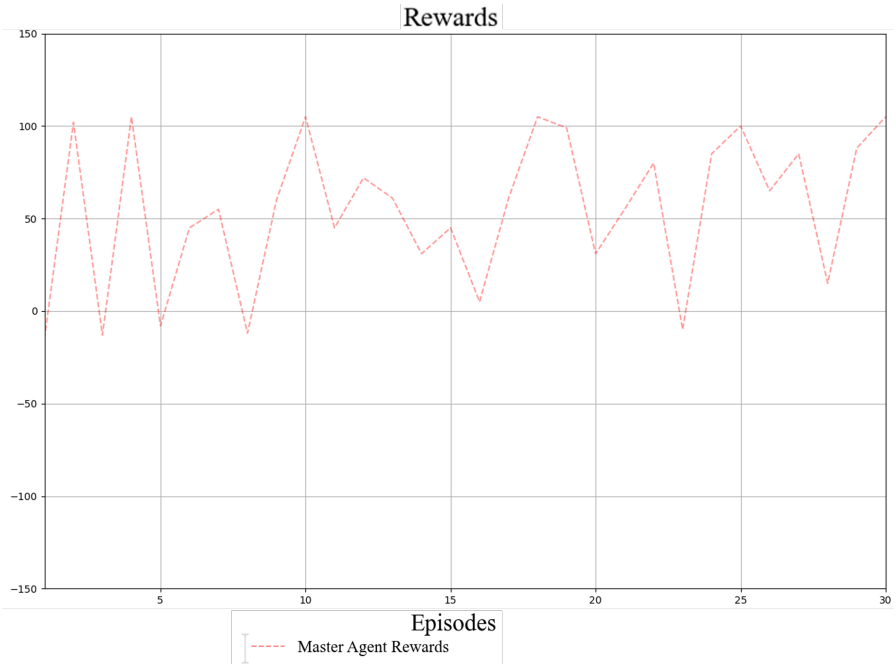

Figure 4.26: Results of the re-trained PPO master-agent with the DEM environment.

4.5.3 Summary

The described approach of tackling the multi-actuation transportation task uses the developed pre-training methodologies of the iterative learning schedule with multi-complex environments and also the hierarchical RL framework. Dividing the whole task into different levels of complexity and assigning them to sub-tasks, facilitates to solving the entire complex problem. Since training the agent from scratch for the multi-actuation transportation task with the DEM environment is not feasible due to the long computation times, the developed methodologies results in a suitable learning behavior and reduce the training time.

The HRL framework of the multi-actuation transportation task consists of two levels with one master-agent, two low-level sub-agents and two sub-controls. The actual control of the position of the actuator-units is shifted to a low-level control with individual sub-agents for each unit. These sub-agents observe the machine constraints and move the unit into the target area. The novel incorporation of sub-controls available for both units, require not training time and consist of hard-coded software. These controls simply moves the units forwards and therefore

transports the parcel in the desired direction. The master-agent observes the whole environment and orchestrates both actuator-units to successfully transport a randomly placed parcel along the film. The used environment for this task is an enhancement of the developed digital twin of the PSM of single-actuation transportation task of the previous section. The whole training of the master and sub-agents with the pre-training with simplified environments and the re-training with the complex DEM environment achieves promising results. After the training, the agent is able to control the machine to successfully transport a parcel with multiple actuator-units. The developed and contributed methodologies of the pre-training and the HRL framework are adaptable to any other RL problems, but are especially useful when training agents in computationally intensive environments.

4.6 RL-PLC Implementation

Even with new developments of industry 4.0 and keywords like Industrial Internet of Things (IIoT), edge computing, cloud computing, etc. the PLC still forms the backbone of modern CPS [205]. However, PLCs of different brands are programmed with proprietary programming languages and only offer limited possibilities for incorporating modern machine learning libraries. But using machine learning and especially RL could improve the performance of CPS drastically and is thus under investigation [206]. That is why, a common trend is to shift the programming and execution of any kind of machine learning to edge or cloud computing devices. These devices offer the possibility to communicate with the desired application and have sufficient capacities to train and deploy algorithms. Cloud services in particular are cautiously accepted in the industry due to data security, privacy and governance issues [206]. One approach to overcome this problem is to shift the computing within the machine network for example to edge computing or as in [207], while directly communicating via the fieldbus level. However, these solutions could face disadvantages regarding relatively fast processes due to the communication delays of the network. Thus a machine learning approach close to the main processing unit, PLC respectively, is desired. To name two well-known examples, Siemens developed the neural processing unit [208], a special hardware to deploy DNN on the PLC side and Beckhoff developed the TF3810 [209], a special machine learning library module. These approaches can only deploy trained networks, support only certain programming languages and do not offer possibilities to train with process data etc.

Therefore, training of advanced RL approaches with real environments controlled by PLCs is not developed, yet. Furthermore, there is no comprehensive approach on the PLC level to deploy trained RL agents in real-time i.e. without required slow communication to external components and online training of RL agents. That is why this approach is developed to close the gap between existing machine learning approaches and environments and data on the PLC

level.

To implement machine learning and reinforcement learning directly on the PLC side and use state-of-the-art libraries e.g. Python with Pytorch, a RL-PLC implementation is explained in the following. To enable machine learning on the PLC, appropriate hardware is required. The hardware must be sufficiently powerful, which strongly depends on the desired application and machine learning approach. The described approach is applied to Siemens IPCs, but basically adaptable to any other soft-PLC.

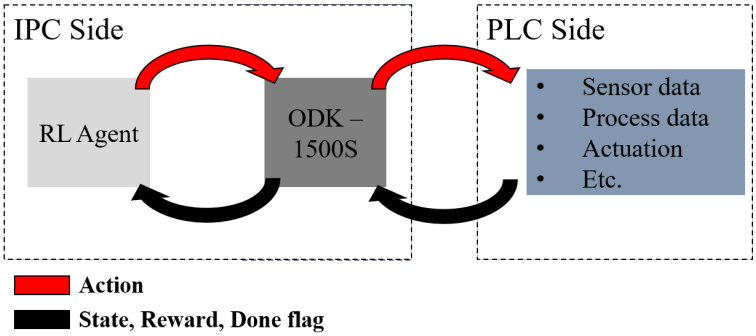

Figure 4.27: Communication framework of the RL-PLC implementation.

Generally, the approach is based on the communication structure shown in Figure 4.27, where a RL agent communicates with a ODK (Open Development Kit)-1500S object on the IPC side and then exchanges data of the application with the PLC side. The data can consist of sensor input data, process data, any kind of actuation commands, etc. The encoded state information are gathered on the PLC side and propagated trough the communication structure to the RL agent. Based on the policy of the agent, an action is chosen and returned back. Besides, additional information like the reward signal or possible done flags are transmitted.

The Siemens ODK establishes the intercommunication of the IPC side, usually based on a Windows operation system, with the PLC side. Therefore, Siemens ODK is used to generate files and functions executable on both sides of the system. Functions, written in C++ are thus converted into function blocks and accessible in PLC run-time in real time mode [210]. However, Siemens ODK-1500S only supports the Siemens 15xx CPU family. The RL agent is trained and deployed in a Python interpreter, the Siemens ODK is programmed in C++ programming language and allows to directly communicate with the PLC. Therefore, a connection between Python↔ODK↔PLC is investigated.

Developed functions, written in C++, allow the data exchange between both systems while generating and manipulating handshake and communication files. General, multiple approaches could be developed to manage the internal communication, but in this work text-files were used to exchange the state, action, etc. information. One text-file contains the state, reward and possible done-flags and on the other one describes the chosen action. On the one hand, the files are read by the agent to determine the current state of the PLC, the PLC reads the chosen action

of the agent and advances to the next state accordingly. On the other hand, the PLC writes the current state, reward and done flag while the agent does the same with the chosen action. The reading and writing is triggered by the PLC but outsourced to ODK and executed in C++. The entire flowchart of the RL-PLC implementation is depicted in Figure 4.28.

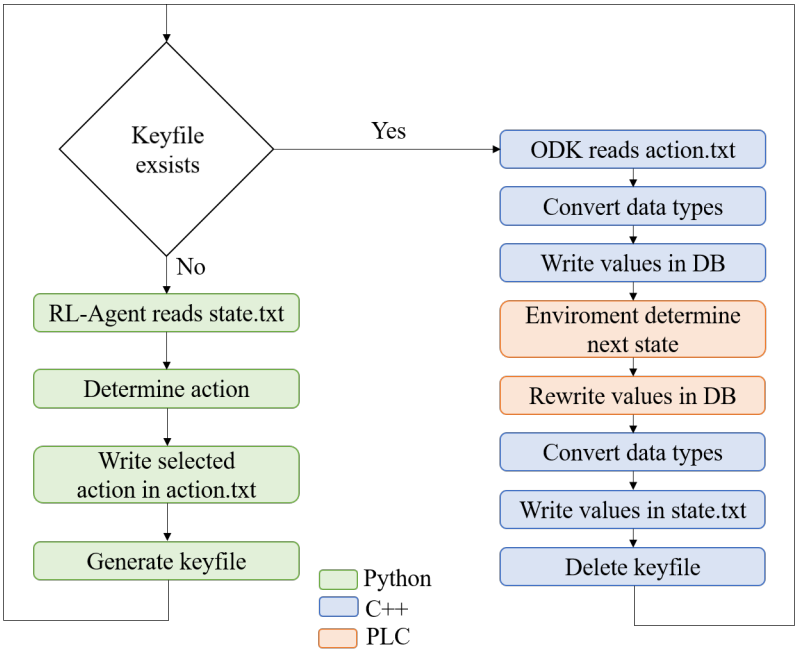

Figure 4.28: Flowchart of the RL-PLC implementation.

The check *Keyfile exists* is used as a handshake file and guarantees consistent data. The agent generates the keyfile after the action has been chosen and the PLC deletes it after it has completely performed the action and wrote the state file. To ensure the correct data-type in the PLC a conversion is conducted and the data is saved into Data Blocks (DB). By following this developed flow-chart, RL can be applied to Siemens PLCs while using state-of-the-art machine learning libraries. The exchange of data is completely performed by the IPC itself so that no connections to third party devices are required. The approach ensures a fully secure operation of machine learning regarding data security. Generally, all kinds of RL algorithms are implementable and the amount of exchanged data is only temporally limited by the conversion- and manipulating-time of the text files.

Using a trained agent that has been previously trained by any type of simulation model is feasible with the described approach. It can also be applied on training from scratch or even a re-training of RL agents with the PLC. However, in contrast to train with simulation models, the PLC interacts with certain hardware e.g. servo drives, brakes, etc. in real-time. That is why it is imperative not to interrupt the PLC runtime while waiting for new actions to avoid possible damage or injury. Depending on the actual implementation of the RL algorithm, a policy update causes a certain communication delay which is not acceptable for highly dynamic environments.

Therefore, mid-episodic updates of the policy are inappropriate and updates are only valid at the end of episodes when the real environment is in a secure state.

4.6.1 Example Training with the IPC-RL Implementation

To show the general applicability of the developed RL-PLC implementation a simplified PSM task is trained on a Siemens IPC. Therefore, an environment based on a three-dimensional grid in the size of the PSM impact area with (L x W x H) 400 x 200 x 100 elements is developed. The task describes the movement of the actuator-unit from a random position to a target position in this discrete grid. The state is defined as a vector of three elements representing the three-dimensional grid. The action-space is defined as six discrete actions resulting in a movement to subsequent elements in positive or negative direction of one axis. The reward is defined as the ratio of the current Euclidean distance between the actuator unit and the target and the initial distance between the two at the beginning of the episode. If the actuator-unit reaches its machine boundaries, the episode will be terminated and the agent will receive a reward of -200. If the agent moves the actuator-unit and reaches the target position with a margin of ± 2 elements, the agent will receive a reward of $1,000$. The environment is directly programmed in Python and simulated on the PLC as well. The environment of the PLC additionally incorporates some noise which imitates deviations in the state caused by the communication delay between the Python-PLC data exchange. Considering that the environment on the PLC interacts with real hardware in accordance with a certain movement time, the iterative learning schedule with the multi-complex environments is applied here, too.

The iterative learning schedule is applied and determines the following hyperparamters for the used PPO agent. The PPO agent consists of one hidden layer with 64 neurons for the actor and critic network and the ReLu activation function between the layers. The output layer of the actor network is equipped with the softmax and the critic network with the linear activation function. The learning rate of the PPO master-agent is set to $\alpha_{\mathrm{lr}} = 2\mathrm{e}{-3}$, the clipping factor to $\epsilon_{\mathrm{clip}} = 0.2$ and the policy update is done over $K_{\mathrm{epoch}} = 4$ epochs. The pre-training of the PPO agent with a maximum number of $1,000$ steps over $9,000$ episodes in the Python environment is shown in Figure 4.29.

The difference of the Python and PLC environment consists of a noise signal added to the actions which leads to slightly different states. Small deviations of the environments and in particular while deploying pre-trained agents on real hardware can have drastic consequences. As explained in Section 2.1 the usage of neural networks as function approximators and stochastic control policies creates a black-box behavior which is impossible to predict. Therefore, to minimize this problem the re-training allows the agent to generalize and behave accordingly even in a slightly different environment. To create a baseline, the pre-trained agent is deployed on the PLC environment for $1,000$ episodes which is shown in Figure 4.30. It can be seen, that

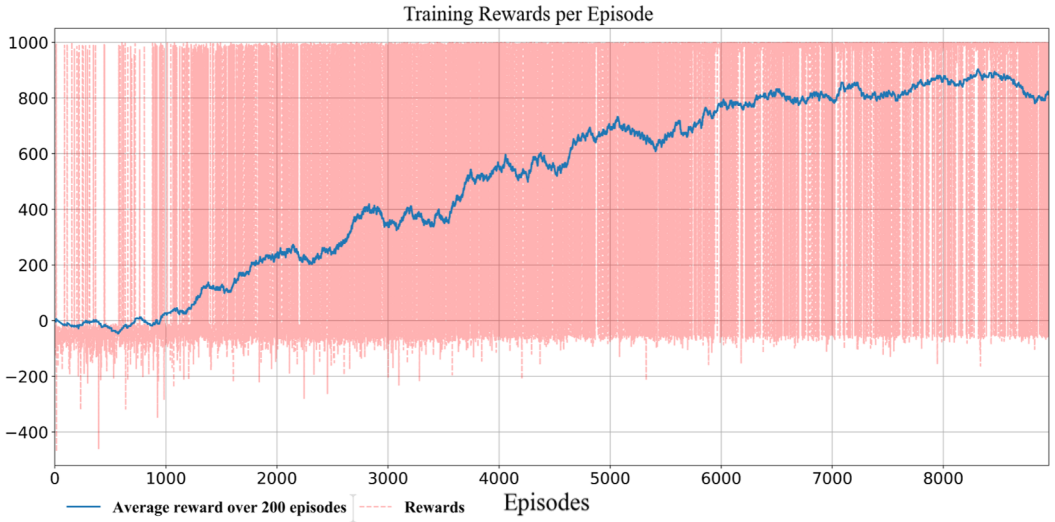

Figure 4.29: Pre-training with the RL-PLC implementation.

the agent is not able to perform well in this environment and never reaches the target position.

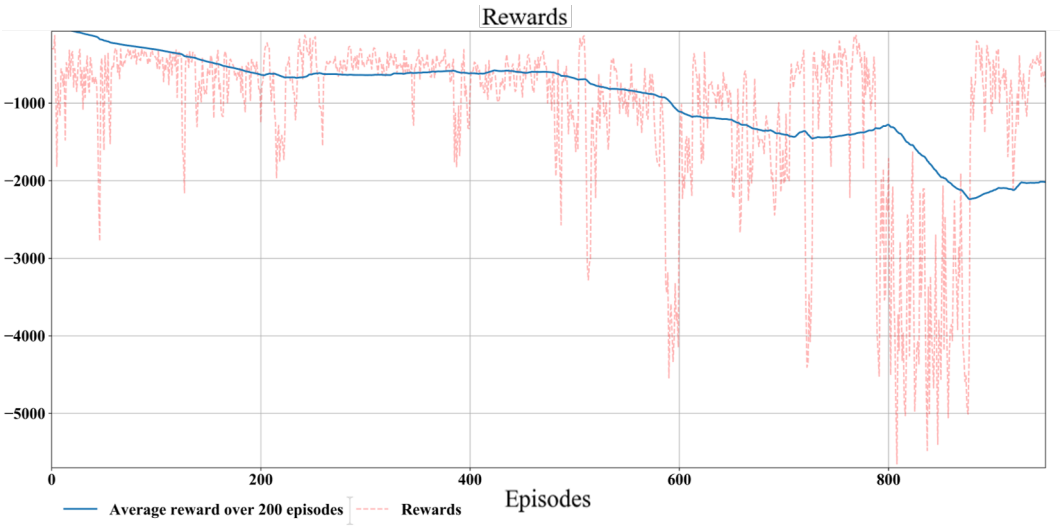

Figure 4.30: Baseline results of the pre-trained PPO agent with the PLC environment.

To increase the performance of the PPO agent a re-training is conducted with the developed RL-PLC implementation. By keeping the hyperparameters similar, but exchanging the environment from Python to the PLC environment, the agent is re-trained again for $10,000$ episodes. In Figure 4.31, the entire reward course is shown, and after approximately $3,000$ episodes the agent can not increase the performance anymore. Further, it is shown that the agent poorly behaves in the beginning of the episodes and then gradually increases the rewards.

Since the PLC environment is made just with a simulation on PLC there is nearly no time required for advancing to the next state. In real applications with any kind of periphery, the execution of actions could take a certain amount of time and re-training with this large number of episodes is not feasible. However, the results after the pre-training, tested again with $1,000$

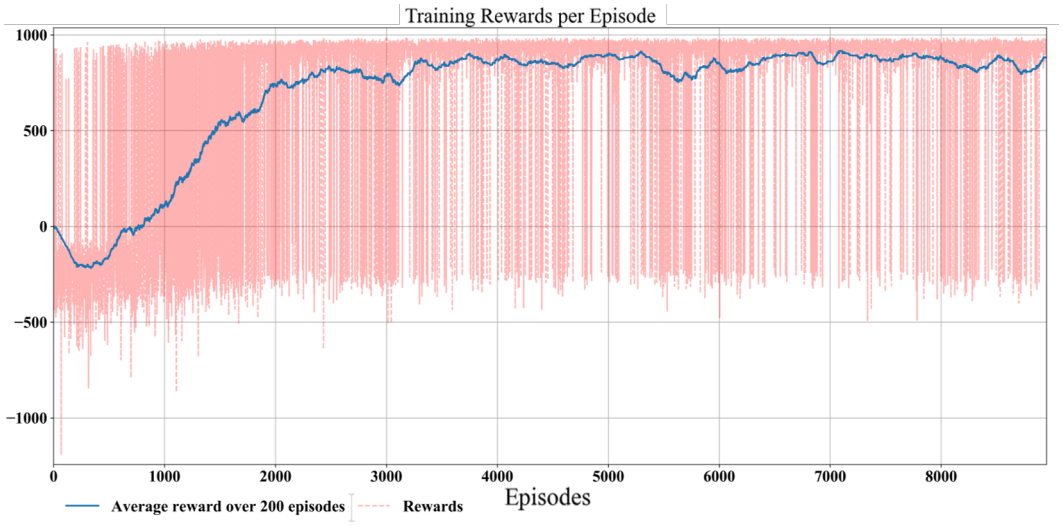

Figure 4.31: Results of the re-trained PPO agent with the PLC environment.

episodes, are shown in Figure 4.32. There, the agent significantly increases the performance compared to the previous baseline and the re-training reduces unintended action choices of the agent. After the re-training the agent is able to reach the target in 96% of the episodes and another re-training could increase the performance even more.

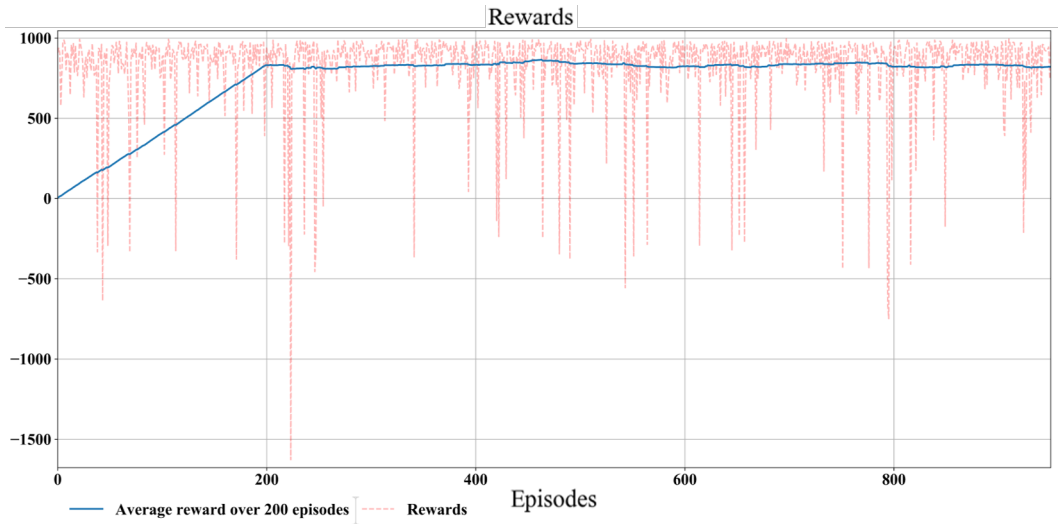

Figure 4.32: Results of the re-trained PPO agent with the PLC environment.

This simple example of the developed RL-PLC implementation is generally adaptable to other applications and shows the applicability of the developed approach. Until now, this novel approach offers the only possibility to deploy RL agents which run in Python to directly train with real machines operated by Siemens IPCs. It allows the deployment and asynchronous update of RL agent dealing with PLC data without external components. Future work on running the PSM with RL and real hardware will end up in the three stages of training, starting with

the pre-training with the simplified environment $\rightarrow$ DEM environment and finally re-train and deploy the agent with the RL-PLC implementation on the real PSM.

5

DEM Parameter Optimization

In this chapter, the prominent DEM optimization problem is tackled with the coupled RL-DEM approach and digital twins of the corresponding calibration tests. Nowadays, the comparison of multiple physical calibrations tests with DEM simulation models attempts to solve the problem of the inherent ambiguous solution of DEM simulation based on a batch of different parameters. However, this high dimensional calibration problem requires a sophisticated optimization approach to reduce the number of iterations of comparing real with simulated results and increase the quality and accuracy of the entire optimization approach. Therefore, in this optimization approach, a RL agent is used to efficiently find suitable DEM material input parameters to then realistically simulate desired materials.

A novel defined MORL approach concerning two objectives iteratively optimizes the material parameters of the target material by observing suitable DEM simulation environments [211]. The desired target material is automatically evaluated with the developed the automated mobile calibration unit to enable precise results and a high level of repeatability. Since the computation of DEM simulations requires significant resources and time, the number of iterations should be kept as small as possible. Therefore, the approach incorporates the developed methodology of the pre-training strategy for DEM parameter optimization and thus significantly reduces the necessary optimization runs and thus the required computation time. The pre-training strategy uses a simplified MDP where the action is the change of the states itself and thus allows agents to learn the task rapidly. Additionally, the agent learns the reward function which combines all non-quantifiable material parameters combined into Θ_0 of a proxy material. In this pre-training the agent learns to solve the optimization with the desired targets, but interacts with a proxy material which results in a fast computation. By that the trained agent is then able to transfer its

knowledge to other materials with a short re-training process.

The combination of gathering the material properties with the developed automated mobile calibration unit and optimizing the material parameters with the almost fully automated calibration procedure creates a novel and remarkable DEM parameter optimization approach.

5.1 Calibration Procedures

As already briefly explained in Section 2.3, the calibration of DEM parameters yields to matching results of the physical tests and surrogate models in the simulation. By observing target key parameters and varying the input DEM material parameters at least one suitable material parameter set is to found.

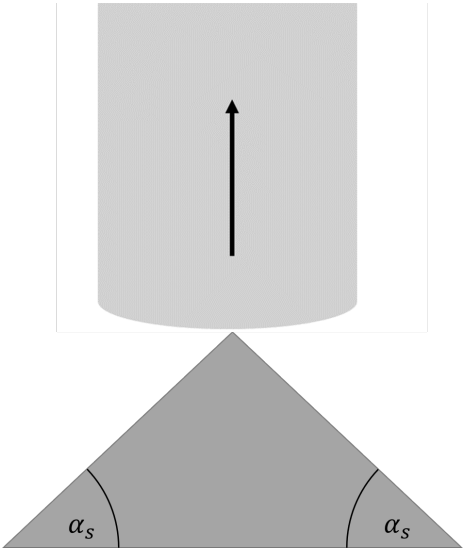

Figure 5.1: Static AoR measured using the lifting cylinder experiment [211].

In this work two target key parameters, the static and dynamic AoR, which essentially describe the bulk material behavior, are investigated. The static AoR, commonly used to characterize the flowability of bulk materials [66], is determined with the lifting cylinder experiment as depicted in Figure 5.1. There a hollow cylinder is filled with the desired material and while continuously lifting the cylinder the AoR is formed by the resulting heap. Then the static AoR α_{s} is measured between the horizontal plane and the material surface. Depending on the specific material behavior this angle varies $0° \leq \alpha_{\mathrm{s}} \leq 90°$.

The dynamic AoR, the second target key parameter, is obtained by using the rotating drum experiment, shown in Figure 5.2. There a horizontal cylinder, previously filled with the material, rotates with a slow constant rotational velocity. The resulting dynamic AoR is then measured between the horizontal plane and the material surface. To ensure an evaluable AoR, the diameter of the rotating drum and the speed are adjusted considering the researched material. Ideally, the speed and the diameter are adjusted to obtain a dynamic AoR in the range of $20° \leq \alpha_{\mathrm{d}} \leq 50°$.

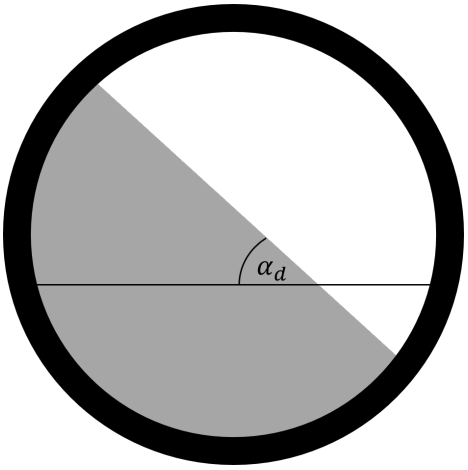

Figure 5.2: Dynamic AoR measured using the rotating drum experiment [211].

To conclusively characterize the material properties it is necessary to evaluate the two AoR key values and a third measurement. The inclined plane as the third measurement is used to obtain the coefficient of friction between the material and relevant surfaces. The inclined plane tester, shown in Figure 5.3, determines the coefficient of static friction μ_s. Generally, this coefficient describes the relation between the friction force and the normal force of two not moving bodies with

$$\mu_\mathrm{s} = \frac{F_\mathrm{f}}{F_\mathrm{n}}, \tag{5.1}$$

$$\mu_\mathrm{s} = tan(\alpha_\mathrm{max}). \tag{5.2}$$

The angle α_max is measured by placing the desired material on the relevant surface and lifting one site until the specimen starts sliding downwards.

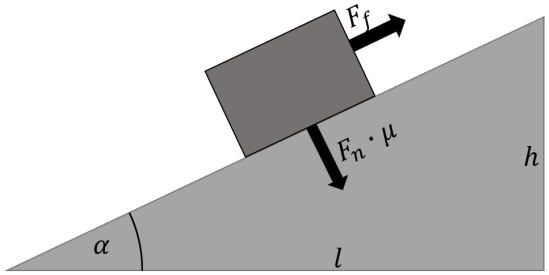

Figure 5.3: Determination of the coefficient of friction with the inclined plane test [211].

To finally validate the results after the optimization process, a modified draw-down test according to [64], is conducted and the results are compared to the simulation. As shown in Figure 5.4, this validation test consists of an upper box filled with the desired material, a flap that opens to let the material flow in the two lower layers, the middle platform and the lower box.

This test allows to measure the resulting shear angle α in the upper box and the AoR β of the poured heap on the middle platform.

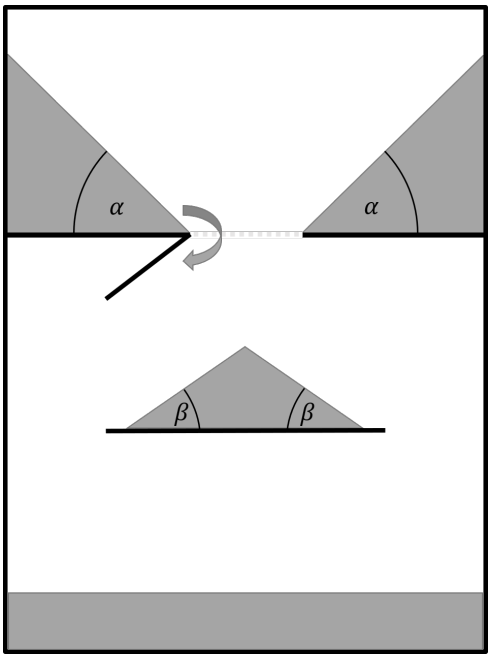

Figure 5.4: Validation experiment with the draw-down test [211].

To avoid statistical weaknesses the aforementioned tests are performed several times. Additionally, standardized measurement equipment, meaningful descriptions of the experiments and traceable test results improve the calibration quality and are entirely embedded into the developed automated mobile calibration unit.

5.2 Calibration Unit

The developed automated mobile calibration unit is equipped with a variety of different physical which are used for the calibration. The calibration unit is equipped with the lifting cylinder and rotating drum calibration procedure, the inclined plane tester to measure the coefficients of friction, a bulk density measurement and the validation test. Sensor equipment measures the individual target key parameters and a camera system records the tests for possible post-processing. Due to its mobility, the novel unit offers possibility to directly analyze the desired material at site, which brings advantages when the outside temperature or humidity must be taken into account [212]. Assisted with state-of-the-art automation technology, the unit ensures high quality and reproducible measurements.

Figure 5.5 shows the entire calibration unit and with the HMI, which guides the operator through the different calibration tests and allows for an operator-independent calibration. The gathered material data are saved in the unit itself on a flash drive, or directly in a cloud. The

data consist of general material information, measurement results of the different tests and supplementary material. Together with shape and particle size distribution information, these data are transferred to a local machine that accordingly performs the optimization algorithm.

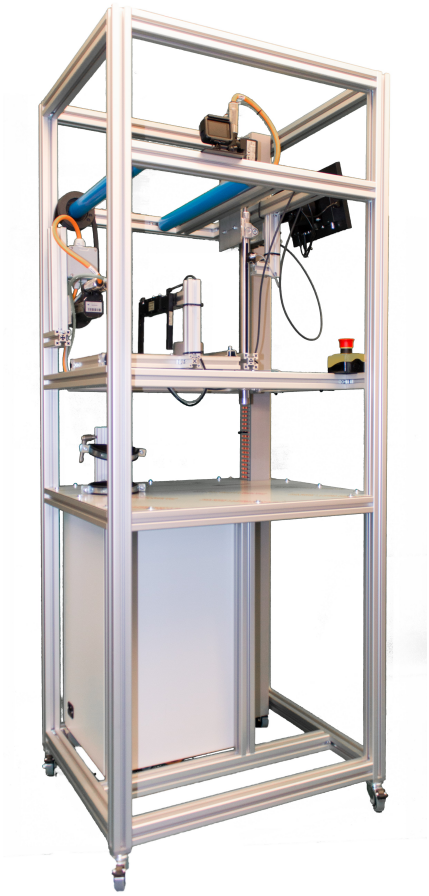

Figure 5.5: Overview of the mobile DEM-Calibration unit [211].

5.2.1 Materials

To show the applicability and performance of the developed optimization algorithm three different bulk materials have been tested and validated by the automated mobile calibration unit. The three materials are plastic granulate, wood pellets and wet sand. They differ in size, shape and general bulk behavior. Since there are no valid materials parameters available, these materials require a proper DEM calibration. As shown in Figure 5.6, the plastic granulate consists of almost spherical particles with a diameter of 3-4 mm and has an average bulk density of $\rho_{\mathrm{b,p}} = 367\ \mathrm{kg/m^3}$. The wood pellets, made of pressed sawdust, have a cylindrical shape with a diameter of 4 mm, a length of 5-35 mm and a determined average bulk density of $\rho_{\mathrm{b,w}} = 481\ \mathrm{kg/m^3}$. With a moisture of $\approx 10\%$ the wet sand is measured with a bulk density of $\rho_{\mathrm{b,s}} = 1,617\ \mathrm{kg/m^3}$ and has roundish particles with an average diameter of 1 mm.

Figure 5.6: Calibrated materials, from left to right: Plastic Granulate; Wood Pellets; Wet Sand [211].

Other important material characteristics are the shape approximation and modelling in the DEM, the size of the particles and the particle size distribution. In the developed calibration procedure the shape approximation is intended to set manually. In the manual setting, the shape can be distinguished between simple spheres, multi-spheres, or superquadrics. Plastic granulate and wet sand are approximated by simple spheres while the wood pellets are simulated using a multi-sphere template consisting of four overlapping spheres in a row. The individual particle size and the particle size distribution are measured with sieves. However, to save computation time and limit the maximum number of particles in the simulation domain, the particle size is artificially increased, which is also denoted as coarse-graining. The level of coarse-graining is defined by the up-scaling factor and described by the scaling law [213]. The up-scaling factor S_{up} needs to be set for different materials and the desired applications, individually. In recent works, the maximum scaling factor has been defined to be 4 to properly simulate the dynamic AoR with a rotating drum [214] or between 3 and 5 using the Hertz-Mindlin contact model [215]. In another work, the original size of the particles has remained and instead the test-rig has been up-scaled accordingly [216]. The approach in [216] has been successfully tested for the determination of the static AoR with an up-scaling factor less than 4. In [217] the influence of the up-scaling factor in the calibration is described. For calibrated parameters, the used up-scaling factor must therefore also be specified.

The shape approximation and the coarse-graining cause a deviation of the bulk density in the simulation and require an adjustment of the particle density. Therefore, a cylinder of a certain size is filled with the desired material and the total mass of the material is then determined in the simulation. The deviation of the mass, considering the ideal bulk density, is measured and the particle density is interpolated and then adjusted. Additionally, to entirely describe the material properties, the COF between the materials and the relevant surfaces i.e. stainless steel as ground material and painted surface inside the rotating drum are measured. A summary of the relevant material properties including the measured target AoR of the static and dynamic test scenarios which are used in the simulation are shown in Table 5.1.

Table 5.1: Material simulation properties and target AoRs.

Parameter	**Symbol**	**Plastic Granulate**	**Wood Pellets**	**Wet Sand**
Particle shape	-	Sphere	Multi-Sphere	Sphere
Up-scaling factor	S_{up}	1.3	1.0	4.0
Particle radius	r_{p}	$3.9 - 5.2 \cdot 10^{-3}$ m	$4 \cdot 10^{-3}$ m	$1 - 4 \cdot 10^{-3}$ m
Bulk density	ρ_{b}	$579.03 \frac{\mathrm{kg}}{\mathrm{m}^3}$	$664.7 \frac{\mathrm{kg}}{\mathrm{m}^3}$	$3104 \frac{\mathrm{kg}}{\mathrm{m}^3}$
COF (stainless steel)	μ_{st}	0.383	0.325	0.561
COF (painted surface)	μ_{ps}	0.624	0.582	0.485
Target static AoR	α_{gs}	25.7°	36.6°	60.7°
Target dynamic AoR	α_{gd}	30.6°	42.8°	40.0°
Tolerance	T_{n}	±1°	±1°	±3°

5.3 Methodology

Solving the DEM parameter optimization problem, by iteratively evaluating multiple objectives yields to solve two major problems. First, by combining multiple objectives, the dimensionality of the problem grows drastically and thus requires an advanced and sophisticated optimization approach. Second, an iterative optimization approach is accompanied by the necessary DEM simulations in every iteration, which results into long computation times. MORL is generally applicable to this kind of problem, but needs to be adapted to properly solve the DEM optimization problem and is therefore methodologically processed in the following multi-objective DEM optimization approach. However, training of a MORL agent from scratch needs too many training episodes and is therefore not practically applicable along with DEM simulations. Therefore, the developed method of the pre-training strategy assists the agent to gain knowledge and thus reduces the entire training time.

5.3.1 Multi-Objective DEM Optimization

The comparison of different real calibration tests with surrogate DEM based models requires a holistic optimization approach which evaluates multiple tests scenarios with regard to an optimal parameter set. Therefore the developed multi-objective optimization seeks to find an optimal material parameter set to match the two target AoRs and is based on MORL multi-objective reinforcement learning as explained in 3.2.5. Based on the mentioned scalarization approach, a single policy MORL for $m_o = 2$ objectives, incorporating the A2C RL algorithm is developed. The approach is, however, also extendable for more objectives or other RL algorithms. The entire MORL-A2C framework for the DEM optimization with both environments, the individually obtained angles and the scalarization is shown in Figure 5.7.

In the MORL-A2C framework one A2C agent simultaneously interacts with two DEM

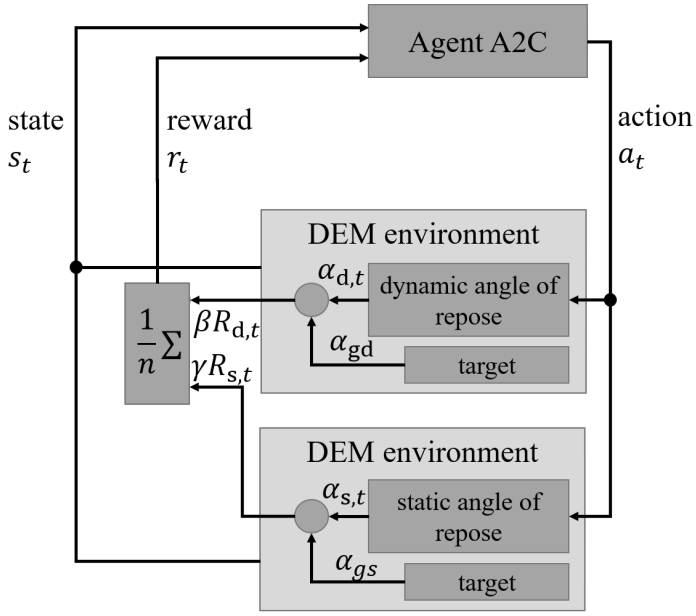

Figure 5.7: MORL-A2C framework for DEM optimization.

environments represented by the digital twins of both AoR experiments, but which are combined into only one MDP. The discrete state s_t represents the current material properties as a vector of the four concerned properties which are the coefficient of friction, - rolling friction, - restitution and the cohesion parameter. According to the current state, the agent decides an action a_t which represents the discrete change of one parameter of the state vector. This positive or negative parameter change limited with a given step-sizes yields consequently in a vector of eight elements. To allow the agent to find a suitable solution for multiple objectives, the reward signal r_t is composed of a rewarding which transposes both objectives into one scalar reward. To this end, the linear scalarization is given by

$$r_t = \frac{1}{n}\sum_{i=1}^{n}(\omega_i r_i), \tag{5.3}$$

where the individual rewards of the objectives $r_{i,t} \in \{r_{\mathrm{d},t}, r_{\mathrm{s},t}\}$ are individually computed with

$$r_{\mathrm{d},t} = \left(1 - \frac{|\alpha_{\mathrm{gd}} - \alpha_{\mathrm{d},t}|}{\alpha_{\mathrm{gd}}}\right)^2, \tag{5.4}$$

$$r_{\mathrm{s},t} = \left(1 - \frac{|\alpha_{\mathrm{gs}} - \alpha_{\mathrm{s},t}|}{\alpha_{\mathrm{gs}}}\right)^2. \tag{5.5}$$

By adjusting the weighting factor ω_i individual objectives can be prioritized. In the DEM optimization approach, the goal of the agent is to find suitable parameters which match the measured target angles of the static AoR α_{gs} and the dynamic AoR α_{gd}. Therefore, the reward is based on evaluating the deviation of the obtained static and dynamic AoRs $\alpha_{\mathrm{s},t}(\Theta), \alpha_{\mathrm{d},t}(\Theta)$ in

the simulation to the measured targets as well as the boundaries of the material properties. These simulated AoRs depend on the used input parameters and the intrinsic material characteristics (Θ). These characteristics can be manifolded by any dimension and are e.g. the exact shape and size of individual particles. The rewarding is thus determined for each time step by

$$r_t = \begin{cases} 5, & \text{if: } |\alpha_{\mathrm{s},t} - \alpha_{\mathrm{gs}}| \leq T_{\mathrm{n}} \ \wedge \ |\alpha_{\mathrm{d},t} - \alpha_{\mathrm{gd}}| \leq T_{\mathrm{n}} \quad (5.6) \\ \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(\omega_i R_i), & \text{if: } |\alpha_{\mathrm{s},t} - \alpha_{\mathrm{gs}}| \geq T_{\mathrm{n}} \ \wedge \ |\alpha_{\mathrm{d},t} - \alpha_{\mathrm{gd}}| \geq T_{\mathrm{n}} \quad (5.7) \\ -1, & \text{if: } \ \alpha_{\mathrm{s},t} = 0 \ \vee \ \alpha_{\mathrm{d},t} = 0\,, \quad (5.8) \end{cases}$$

where T_{n} is the tolerance of the desired material. The tolerance T_{n} also described in the Table 5.1 are manually defined and describe the accuracy of the repeatability of the measurements of a certain material. In this approach, the weighting factors $\omega_1 = \beta, \omega_2 = \gamma$ are set to 0.5 for both objectives, because both objectives have the same importance. The described rewarding allows the agent to learn to sufficiently move through the state space and find a solution which fits for both objectives equally. If the agent chooses and action which exceeds the parameter boundaries the resulting AoR will be set to zero and the agent will receive a negative reward.

5.3.2 Pre-training Strategy for MORL Optimization

Along with supervised learning, RL is also affected in terms of overfitting and generalization [218]. Usually, to avoid overfitting and enable the agent to generalize, the training of the agent is conducted with a large number of training episodes and incorporating some kind of noise. Similar to supervised learning the training samples are separated into a training and testing set. Hence, at the generalization or memorization ability of agents is achieved when a policy is trained on an initial set of training trajectories and performs well on a set of different testing trajectories [219].

However, if the task is very complex and the environment is represented by a continuous and high-dimensional state space, the required neuronal network structure and thus the training time to generalize well in all sub-spaces of the environment increases drastically. Additionally, while working with real or computationally very slow environments the training time becomes problematic and requires a sophisticated pre-training strategy. Since a pre-training strategy has to be individually set and requires a deep knowledge of the desired process, it is not only described in general, but also demonstrated using the explicit example of the DEM parameter optimization. Nevertheless, the strategy is adaptable and utilizable for any other high-dimensional tasks.

The pre-training strategy is developed to successfully optimize the DEM parameters by using a unified material. This material acts as a proxy, is used to learn the behavior of any other materials and supports the RL agent to generalize well. To counteract the large computation times caused by the long simulation times of the relatively slow DEM solver, the training

is expedited with proper pre-training, in which the agent is trained with a pre-recorded and simplified pre-training data-set. For optimizing the parameters of different materials, the final goal of the RL agent is not to maximize the total return but to reach a final state which gains a maximum single reward. By that, the agent learns to calibrate the reward function itself.

Since the state-space of the optimization problem consists of the current material parameters and the actions are defined as the parameter change itself, the transition probability $P(s_{t+1}|s_t, a_t)$ of the MDP of the DEM optimization problem is simplified with

$$a_t = \Delta s, \tag{5.9}$$

$$s_{+1} = s_t + \Delta s, \tag{5.10}$$

where the action value is equal to the deviation of the state. Thus, the transition probability is always $P(s_{t+1}|s_t, a_t) = 1$ and is easily learnable.

Additionally, the reward function of the MDP of the optimization problem is defined as

$$r_t(s_t, a_t, \Theta), \tag{5.11}$$

where the state-space consists of all quantifiable material properties like Young's module or the coefficient of restitution and Θ summarizes all the intrinsic material characteristics. The reward signal generated by the environment by simulating a certain material is obtained by measuring the deviation between the simulation and the particle behavior of the real target with

$$|\alpha_t - \alpha_x| \rightarrow r_t(s_t, a_t, \Theta), \tag{5.12}$$

where α_t and α_x represent a quantifiable indicator of the particle behavior. If $\alpha_t \approx \alpha_x$ the final state is reached and the reward is maximum. Since the material characteristics combine several discrete and continuous parameters, the learning of the reward function from scratch for various materials is tedious. Therefore, the developed pre-training strategy trains the agent to learn the reward function of a unified material with its characteristics Θ_0. This material can be considered as a proxy for other materials. Especially when changing single material parameters the proxy material behaves fundamentally similar to other materials.

The aim of the pre-training strategy for the DEM parameter optimization problem is to train the agent to find material parameters of the proxy material which behave similarly to the desired material. Therefore, the agent is trained to find the maximum reward considering the reward function of the proxy material $r_t(s_t, \Theta_0)$, but the simulation results are evaluated using the α_x of the desired material. The entire pre-training strategy is shown in Figure 5.8 and consists of

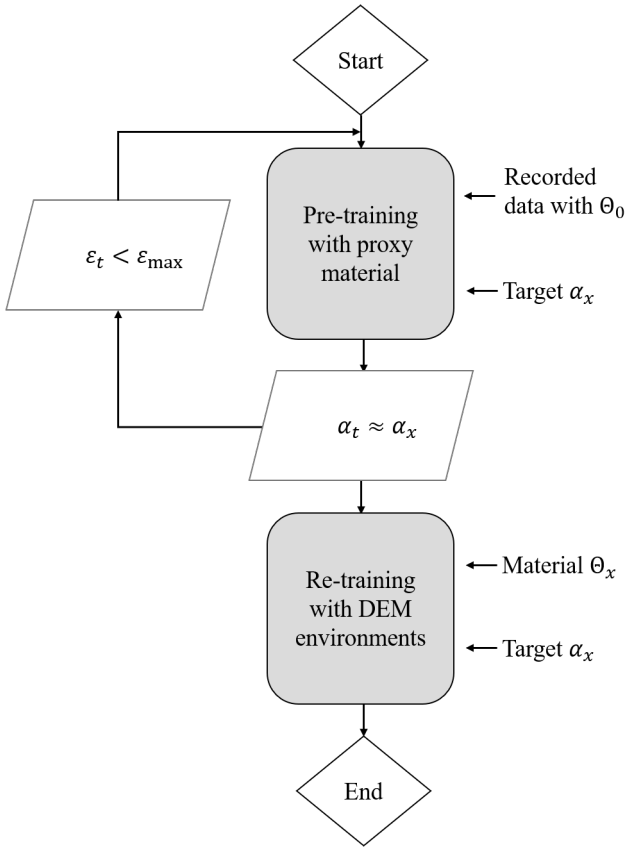

Figure 5.8: Pre-training strategy of the DEM parameter optimization.

the pre-training and subsequent re-training with the actual material characteristics of the desired material.

To speed up the pre-training considerably the behavior of the proxy material with all possible material parameter sets has been simulated completely and recorded accordingly. By iterating all material parameters, the entire state-space $s_t \forall \mathcal{S}$ and the behavior $\alpha_t(s_t)$ of the material are saved in one look-up table. When the agent chooses a certain action a_t, the subsequent state s_{t+1} and the reward r_t, based on the obtained indicator value α_t are picked from the look-up table. That is why the pre-training can be repeated any number of times for different desired materials without the need of simulating the material with the computationally slow DEM model repeatedly.

In the pre-training strategy, the agent starts with a random initial parameter set s_0 and learns to find suitable parameter sets for the proxy material in each episode until the maximum number of timesteps $t_{\mathbf{max}}$ is reached. The agent then repeats the episodes until a suitable material parameter set has been found or the maximum number of episodes $\epsilon_{\max}$ is reached.

After the pre-training, the agent is re-trained with the actual desired material, i.e. represented by the actual particle shape, size and particle distribution. The pre-trained agent is therefore transferred to the re-train task and thus learns the reward function $r_t(s_t, \Theta_x)$ while finding the maximum reward by interacting with the desired material in DEM simulation environments. The

combination of the pre-training and re-training in one single strategy significantly expedites the required training time compared to learning from scratch with the desired material. Ideally, after pre-training, the agent requires only one episode with a few steps to find an optimal material parameter set which is shown in Chapter 5. By choosing a proper proxy material that behaves similarly to the desired material following the expression

$$\frac{r_t(s_t, \Theta_0)}{r_t(s_t, \Theta_x)} \forall s_t \in \mathcal{S}, \tag{5.13}$$

the total amount of re-training is reduced drastically. If the Equation (5.13) becomes 1, there will be no re-training necessary at all.

This pre-training strategy is also adaptable to other optimization problems or any problem concerning a high-dimensional state space. The strategy benefits of learning the reward function by considering only one parameter which concatenates all not quantifiable material characteristics. Instead of using a look-up table, function approximators can be used to represent the recorded data of the proxy material. In case of a big variance of the desired materials, an upstream learning process can additionally decide about the best fitting proxy material.

5.4 Optimization Procedure

The entire optimization procedure including the aforementioned methodologies of the multi-objective DEM optimization and pre-training strategy is enclosed in a single routine following certain steps as depicted in Figure 5.9. It is designed to have a minimum set of input parameters and to almost automatically compute suitable material parameter-sets. The first step presupposes the analysis of the desired material, defining the shape and size of the particles and gathering all required information as shown in Table 5.1. In the second step of the initialization of the RL agent the developed MORL-A2C, explained in detail in the following section, is parameterized with required hyperparameters like the learning rate and the DNN structures, etc. In the third step the material parameters are initialized, starting with the shape approximation, particle sizes and size distribution. Additionally, the bulk density of the shape approximated and coarse-grained material is determined and adapted by a suitable DEM simulation model. In the fourth step, the agent is pre-trained with a recorded pre-training data-set as explained in Section 5.3.2. This developed pre-training allows the agent to learn the fundamental behavior of the material, to generalize and then apply the learned behavior on other similar optimization tasks. Here, the motivation of the pre-training is twofold. First, the training with a recorded pre-training data set significantly saves computation time and also limits the required number of subsequent DEM simulations. Second, it highly generalizes the training results to allow the agent to condition to arbitrary materials with varying properties within reasonable limits.

Therefore, the pre-training data-set consists of data recorded with digital twins and arbitrary material. The material acts as a proxy while all non-quantifiable parameter like the particle shape,

precise size and size distribution are considered as one single parameter Θ_0 which influences the reward function. While the transition probability of MDP of the optimization problem is simplified, the agent only has to learn the reward function in the pre-training.

The choice of the proxy material is irrelevant but must be entirely simulated with all possible input parameter combinations. The digital twins are modeled as surrogate DEM simulation models of the real physical tests. The collected results, i.e. the static and dynamic AoR, are saved into a CSV-file. The file consists of both target AoRs, COF COR, etc. of the relevant and complete input-parameter space shown in Table 5.2. There, four material parameters are varied while the Young's module and Poisson ratio remain constant to keep the state space as small as possible. Considering the given step-sizes of the four parameters, the entire pre-training data-set consists of $10,648$ elements. An increasing number of relevant parameters or reducing the step-size is indeed possible, but requires adjustments of the network capacities.

During the pre-training, the agent learns to match the target AoRs of the analyzed material. First, the agent starts with a random parameter set and while taking actions, it moves through the state space. The action space is defined as the change of one material parameter in the given step size in positive or negative direction. In sum, this results in eight discrete actions. The episodic pre-training ends when reaching the target AoRs and is followed by the re-training with actual material.

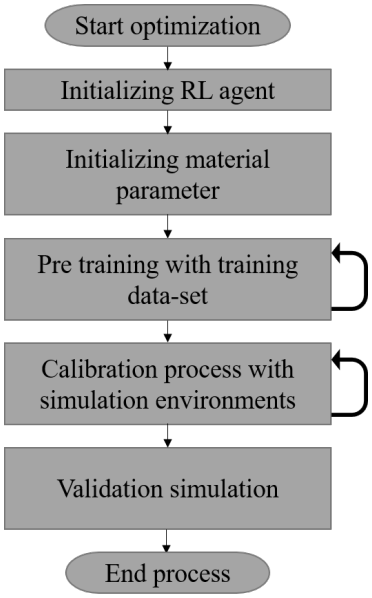

Figure 5.9: DEM optimization procedure [211].

In the fifth step of the optimization procedure, the actual calibration of the DEM parameters is started. The pre-trained agent is retrained by interacting with the developed DEM simulation environments and the desired material. In the re-training the agent has to learn the reward function of the desired material with the material characteristics Θ_x. Further information about

Table 5.2: Material properties used in the simulation.

Material property	**Symbol**	**Unit**	**Dimension**	**Step-size**
Young's modulus	Y	Pa	$5 \cdot 10^6$	0
Poisson ratio	ν		0.45	0
COF	μ_f		$0...1$	0.1
COR	e		$0.1...0.8$	0.1
CORF	k_r		$0...1$	0.1
Cohesion	k_c	J/m^3	$0...1 \cdot 10^5$	$1 \cdot 10^4$

the environments and the rewarding is given in the following sections. Ideally, the agent can find a suitable input material parameter set within just a few iterations which yields to match with the target AoRs. The more similar proxy and desired material and $\Theta_0 \approx \Theta_x$ are, the faster the agent is able to learn the new reward function and to find an optimal set of parameters. If there is no sufficient result after 30 iterations, the calibration will be restarted with a random initial state. At the same time the successful calibration is further validated with the validation test in the next step. In the validation test, the shear and angle of repose are measured and used to prove the quality of the calibration and completes the optimization procedure.

5.4.1 DEM Environments

The optimization algorithm based on MORL is trained to find an optimal solution for two competing objectives. In this approach, these objectives are the static and dynamic AoR, represented by the lifting cylinder and rotating drum experiment. Therefore, both experiments are true to scale and realistically modeled in the DEM as digital twins and functioning as the environments. When the agent conducts an action, both experiments start with the defined material input parameters. The resulting angles are automatically measured and used for the rewarding as well as for the determination of the optimization progress. Both environments are depicted and show the lifting cylinder in Figure 5.10 and the rotating drum experiment in Figure 5.11. The lifting cylinder experiment and simulation use a hollow cylinder with a diameter of 100 mm and a constant lifting speed of 12 mm/s. The drum of the rotating experiment has a diameter of 210 mm, a length of 275 mm and rotates with 10 rpm. However, if needed, the size of the drums and the velocities are adaptable for the desired materials. As previously explained in Chapter 4, the mechanical parts are simplified to speed up the simulation process. To summarize all necessary information for the simulation, Table 5.3 completes the remaining simulation settings and displays the total number of the simulated particles in the simulation domains for the different materials.

The angles to be determined in both experiments depend on the shape of the approximated particles, the particle size and distribution and especially the given material parameters. With

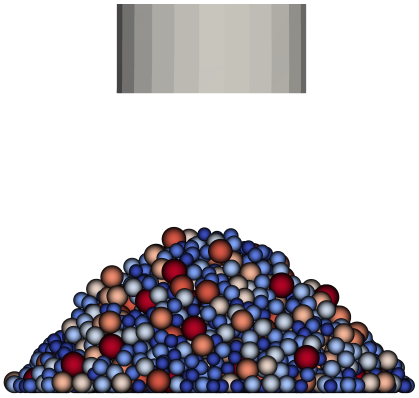

Figure 5.10: Lifting cylinder environment [211].

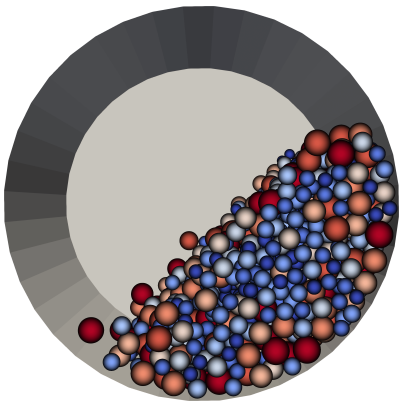

Figure 5.11: Rotating drum environment [211].

every new action, both simulations are repeated and the angles are measured automatically. After a first initialization of the environments, the conditions are saved and used for subsequent runs which significantly reduces the simulation time. Since the DEM is entirely embedded into Python, information of the positions of particles are exchanged and the angle is automatically measured in Python.

Table 5.3: General simulation parameters.

	Static AoR Env.	**Dynamic AoR Env.**
Time step	1e-5 s	1e-5 s
Simulation time	30 s	15 s
Total number of Particles		
Plastic Granulate	3021	2347
Wood Pellets	5640	4852
Wet Sand	43504	22482

The static AoR is determined by measuring the angle of four sides of the heap. Therefore, the particles are represented by their positions and divided height-wise into different groups. The outer particles of each layer are determined and linearly fitted to analyze the AoR of the heap as shown in Figure 5.12. The dynamic AoR in the rotary drum is analyzed equally but only from the two face sides as shown in Figure 5.13.

The validation test which proofs the correctness of the determined input parameters is also simulated by the DEM and compared to the real tests. A manual comparison of the resulting angles is foreseen and further described in the results section.

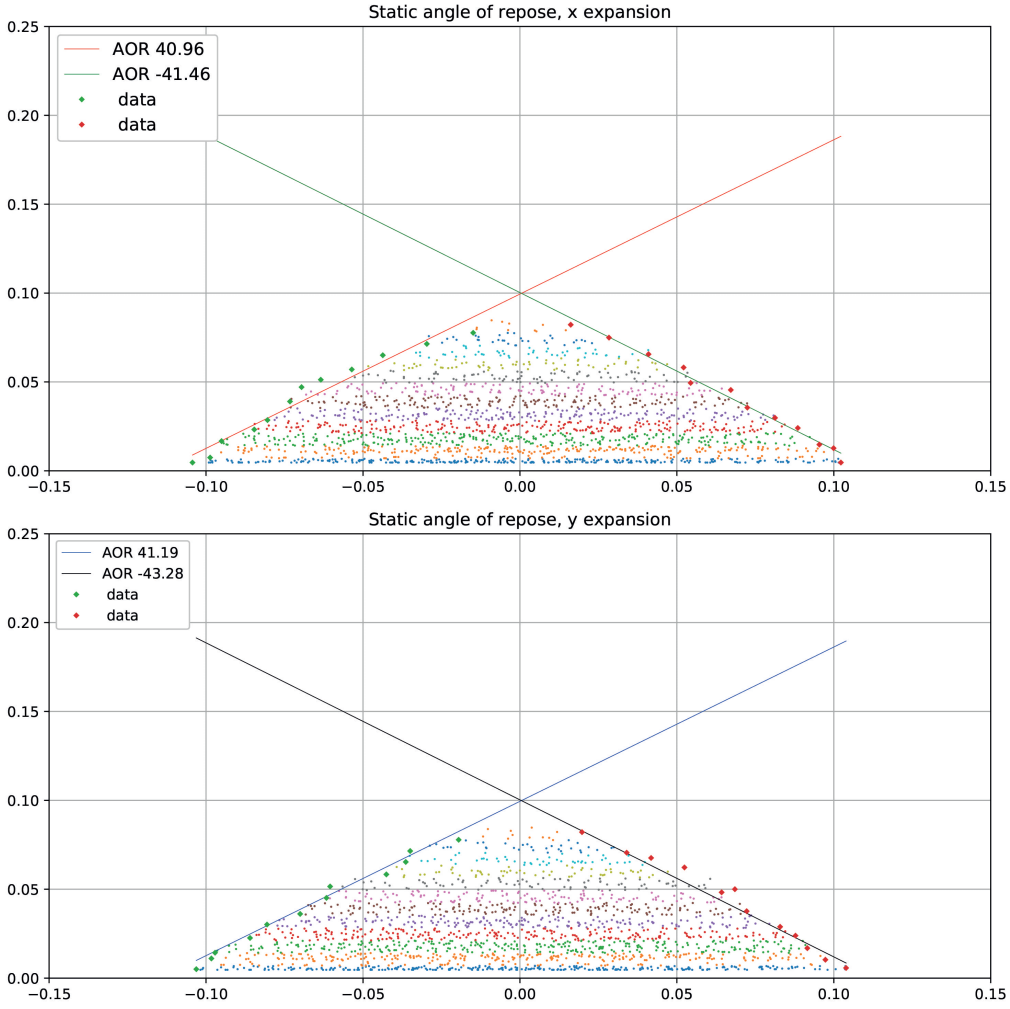

Figure 5.12: Automatic determination of the static AoR [211].

5.5 Optimization Results

The entire optimization procedure of calibrating DEM input parameters is deployed and validated using the three described materials plastic granulate, wood pellets and wet sand. The optimization procedure aims to find suitable parameters for each of these materials with the initialization, pre-training and calibration phase. The procedure is then finally validated with the validation test. The hyperparameter of the MORL-A2C agent and DNNs are equally set for all materials.

The DNN structure is defined by two hidden layers each having 32 neurons and equipped with the ReLu activation function between the layers for the actor and critic network. The output activation function of the actor network is set to softmax to choose a discrete action, the output function of the critic network is set to linear. The discount factor is set to $\gamma_{\mathrm{d}} = 0.9$ and the learning rate is set to $\alpha_{\mathrm{lr}} = 1\mathrm{e}{-3}$ for both networks. The maximum number of episodes in the pre-training is set to 200 and the maximum steps per episode are set to 100. If the agent reaches a final state, i.e. matching the target AoRs or reaching the parameter boundaries respectively, the

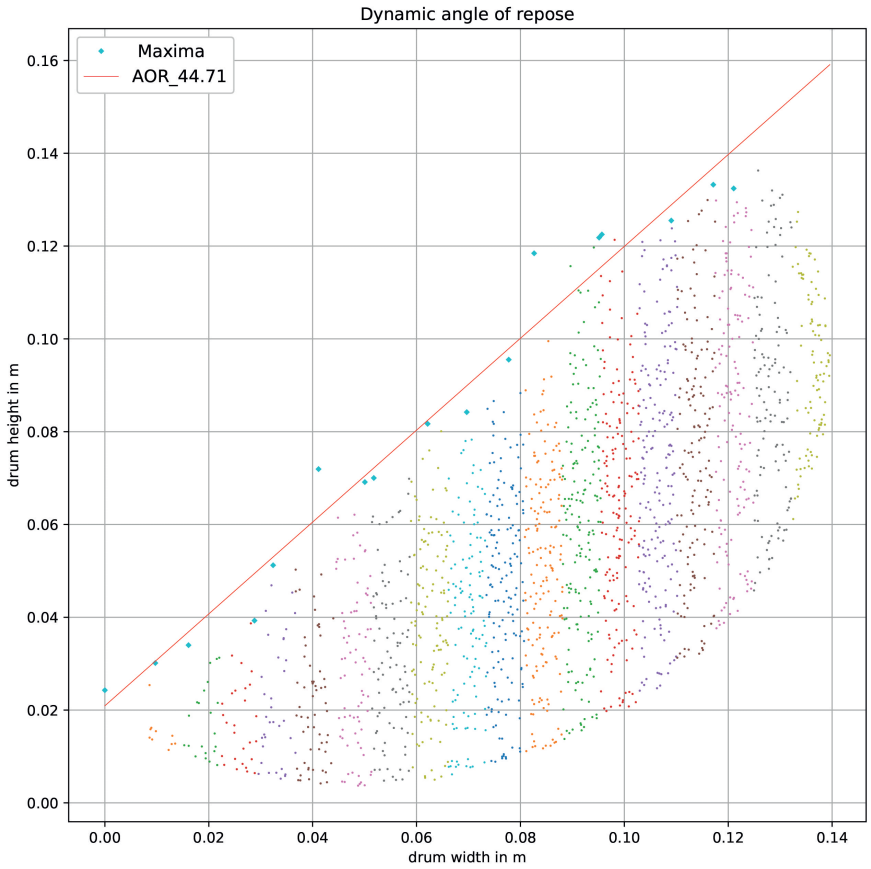

Figure 5.13: Automatic determination of the dynamic AoR [211].

episode ends before the maximum number of steps is reached.

The complete pre-training, solely interacting with the pre-training data-set and the individual target AoRs takes approx. 4-5 min on an Intel Xeon CPU E5-2697 with 32 cores and 64 GB RAM. After the pre-training, the agent has learned to move as fast as possible through the state space and to find a suitable input parameter set that matches the static and dynamic AoR experiment by starting from a random state. Thus the agent has learned the reward function which depends on the proxy material and its intrinsic material characteristics Θ_0. Figure 5.14 shows an exemplary reward course of the pre-training of the wet sand and shows the cumulative reward which gradually increases over the episodes. For all three materials, the agent has been able to sufficiently learn to solve the task and to converge within the 200 episodes.

In the following, the individual results of the optimization of the three materials are discussed. It is important to note that only one optimization run which outputs a single data-set is shown, but nevertheless different parameter-sets could generate similar results due to the ambiguous

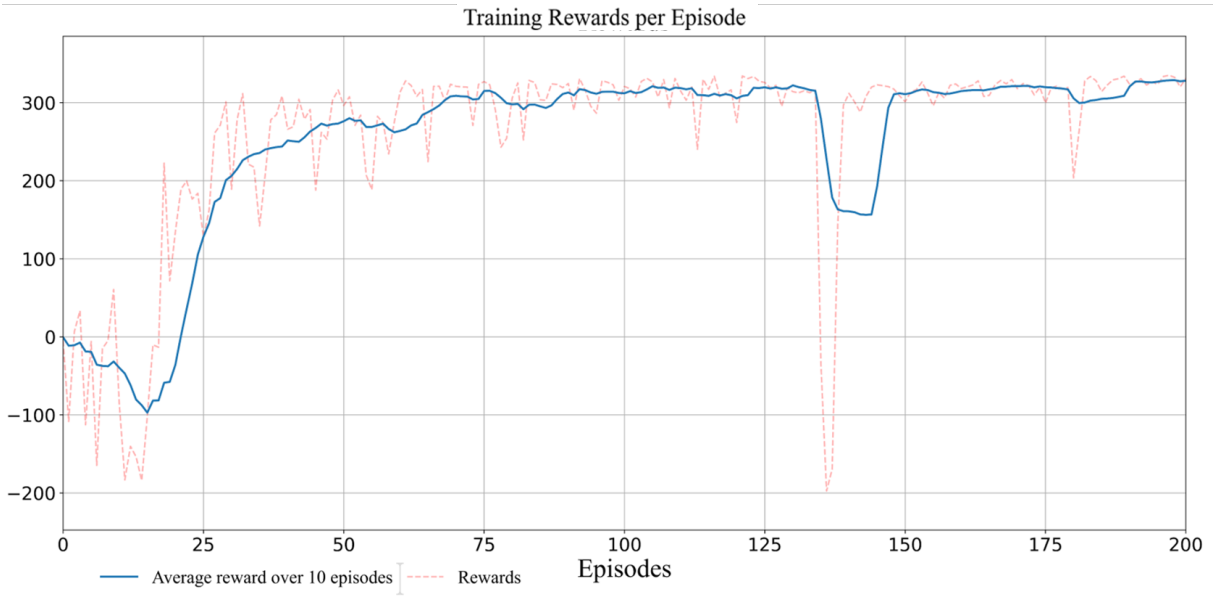

Figure 5.14: Results of the pre-training of the wet sand material [211].

behavior, too. The materials differ in the material characteristics, which are the complex shape, the particle sizes and size distribution. These characteristics are combined into Θ_x and influences the resulting AoRs.

Table 5.4: Results of the calibration optimization of the three materials.

Material property	**Symbol**	**Plastic Granulate**	**Wood Pellets**	**Wet Sand**
COF	μ_f	0.1	0.4	0.1
Cor	e	0.4	0.4	0.2
CORF	k_r	0.4	0.3	0.6
Cohesion $[\mathrm{J/m^3}]$	k_c	0.1	0.1	1.0
Dynamic AoR [°]	α_d	31.4	42.6	42.9
Static AoR [°]	α_s	25.2	35.6	59.6
Dev. Dynamic AoR [°]	$\Delta_{\alpha_\mathrm{d}}$	+0.8	-0.2	+2.9
Dev. Static AoR [°]	$\Delta_{\alpha_\mathrm{d}}$	-0.5	-1.0	-1.1

Plastic Granulate: The optimization of the DEM input parameters of the plastic granulate is shown in Figure 5.15. The result plot depicts the measured AoR of the dynamic test (top plot), the AoR of the static test (middle plot) and the used material parameters (bottom plot). The agent was able to find suitable parameters for the plastic granulate within the tolerance after just 13 iterations. It is apparent that the agent first reduces the cohesion parameter and then adjusts the remaining coefficients. The total number of the required iterations depends on the pre-training and the random initial material parameters. To avoid long computation times due to a poorly chosen initial state, the optimization restarts in a new random state after reaching

30 steps. The pre-training allows the agent to generalize and to find a suitable parameter set in

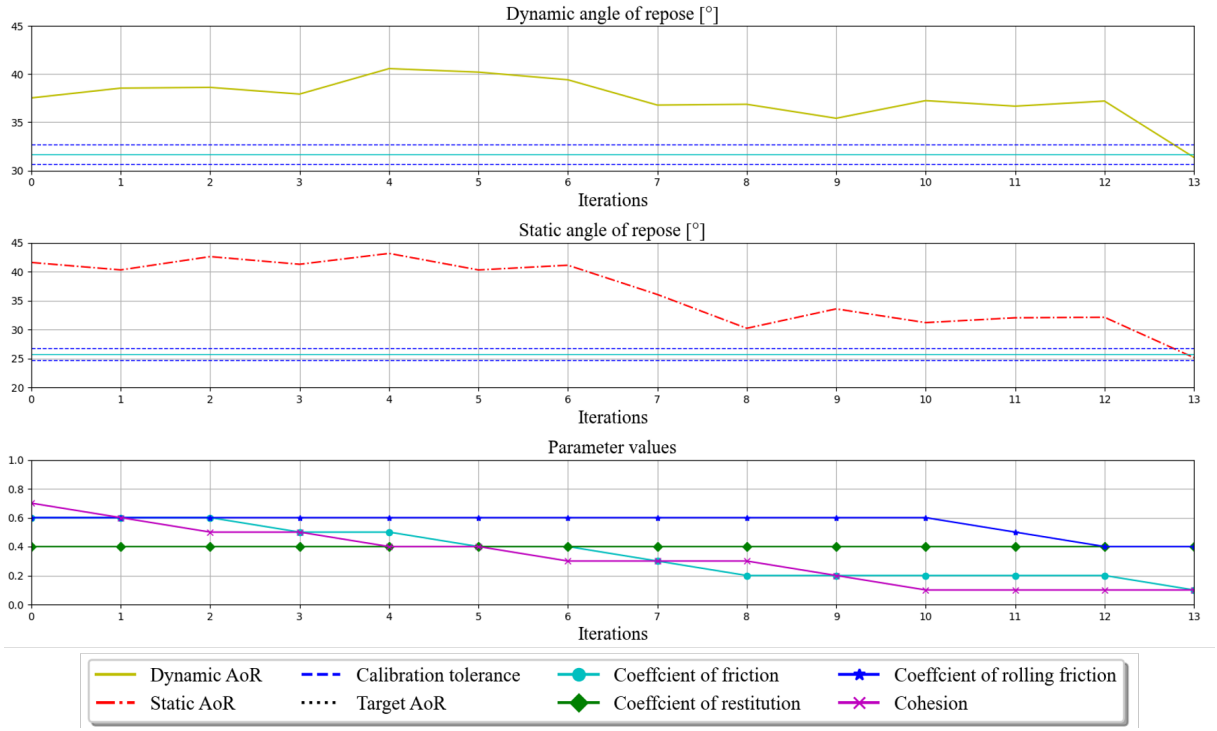

Figure 5.15: Calibration process of plastic granulate and results for the dynamic AoR (top), static AoR (middle) and parameter values [211].

the re-training phase within a few iterations. To underline the essential need for pre-training the agent attempts to optimize the parameters for the plastic granulate without pre-training and is therefore initialized from scratch. As shown in Figure 5.16 the agent failed to find a suitable parameter-set in the given period. The agent randomly selects actions and is exploring the state space. The successful training from scratch leading to appropriate results with the actual DEM environments would take several hundred steps and is therefore not feasible. Hence, the pre-training is mandatory to achieve an optimization of the DEM parameters in a reasonable time.

The resulting parameters for all the materials obtained by the entire optimization are depicted in Table 5.4. The table shows the resulting input parameters, the obtained AoRs as well as the deviations to the target angles. Additionally, all materials are validated by the described validation test and confirm the results. The results of both, the real and the simulated draw-drown test, are visually compared in Figure 5.17. For a quantified evaluation the shear angle α_{sa} and the angle of repose $\beta\mathrm{aor}$ are measured. For the plastic granulate this validation results in just minor deviations and therefore confirms the correctness of the determined DEM input parameters.

Wood Pellets: The optimization procedure of the wood pellets is shown in Figure 5.18. It is important to note that although the pre-training is made with a training data-set simulated with spherical particles, the optimization of multi-sphere wood pellets still leads to appropriate results.

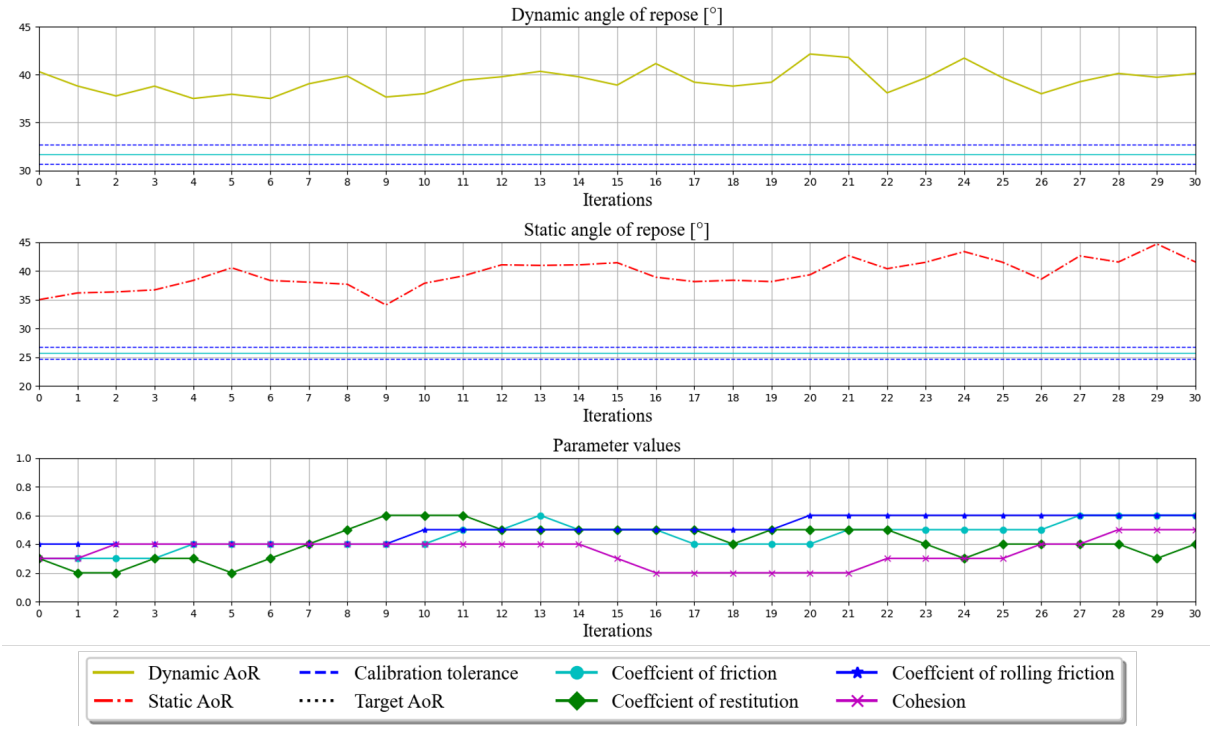

Figure 5.16: Calibration of the plastic granulate without pre-training [211].

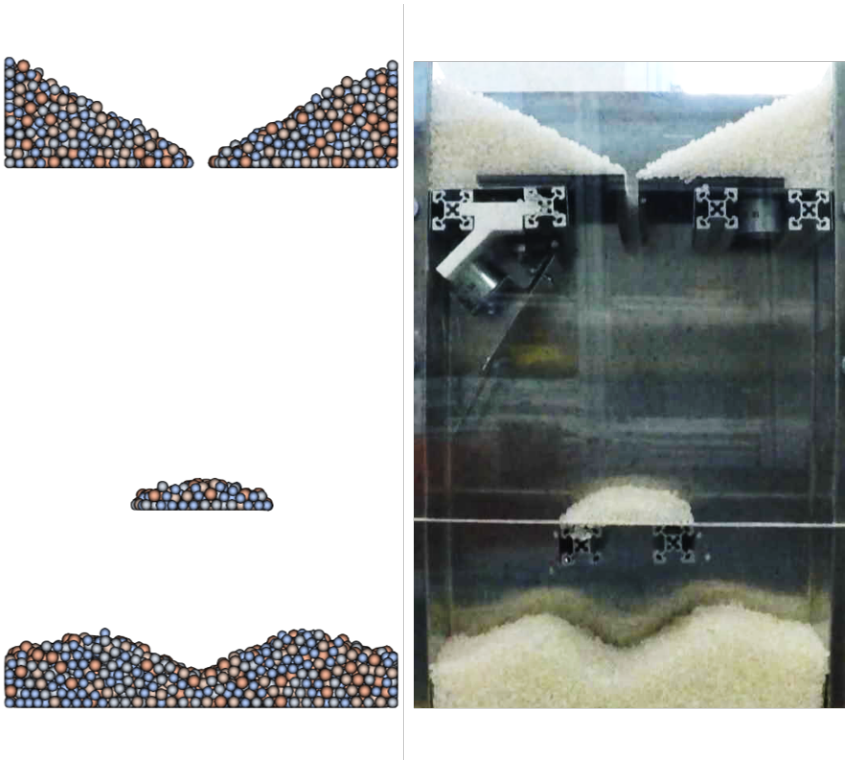

Figure 5.17: Comparison of the validation experiment with plastic granulate (left: DEM simulation, right: Physical test in calibration unit) [211].

Table 5.5: Validation test results: Plastic Granulate.

	Shear Angle α [°]	**Angle of Repose** β [°]
Physical Test	30	34
Simulation Result	29.4	31.2

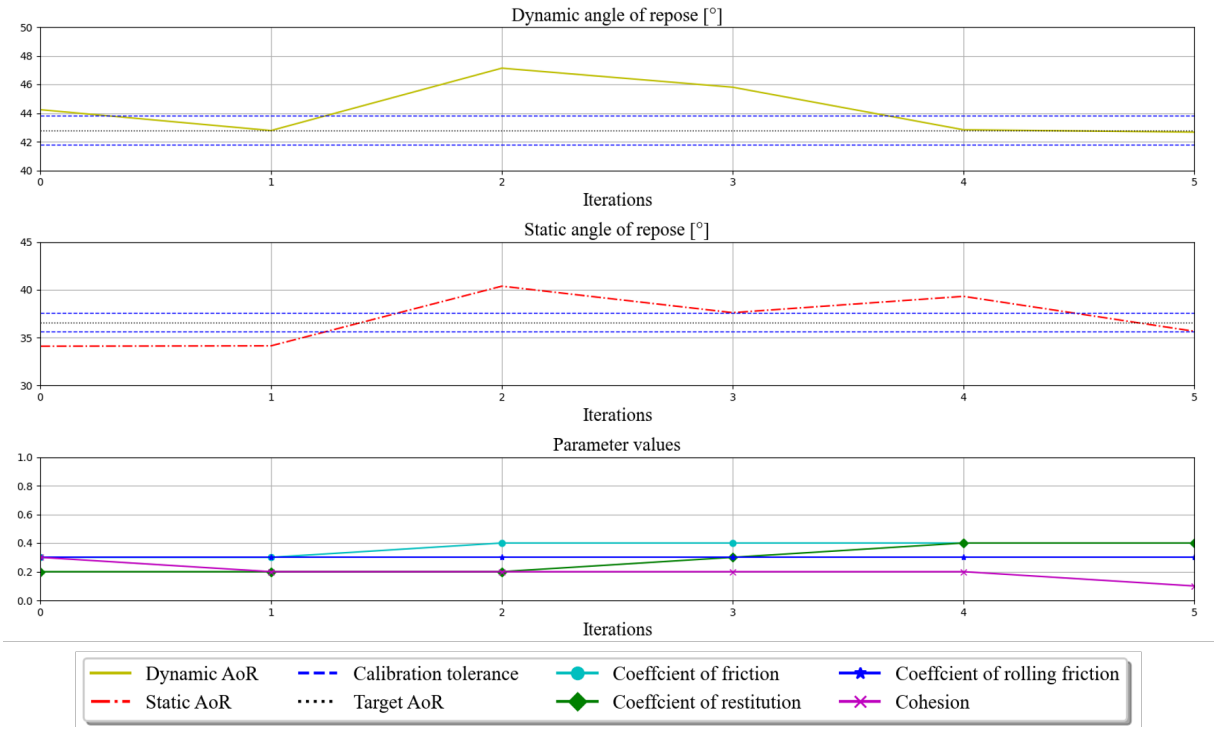

Figure 5.18: Calibration process of wood pellets and results for the dynamic AoR (top), static AoR (middle) and parameter values [211].

The agent is trained to generalize and optimizes the parameters of a different material. This is also underlined by the results in Table 5.4. The validation of the wood pellets, visually shown in Figure 5.19 and also measurable in Table 5.6 proves the results.

Plastic granulate as well as the wood pellets show a cohesionless, real behavior which was also learned by the agent. The used SJKR contact model provides no benefits for simulating such materials but is used to increase the general applicability of the developed approach and calibrate slight cohesion materials, too.

Table 5.6: Validation test results: Wood Pellets.

	Shear Angle α [°]	**Angle of Repose** β [°]
Physical Test	40	54
Simulation Result	41.8	55.5

Wet Sand: The results of the wet sand, representing a standard material with cohesion

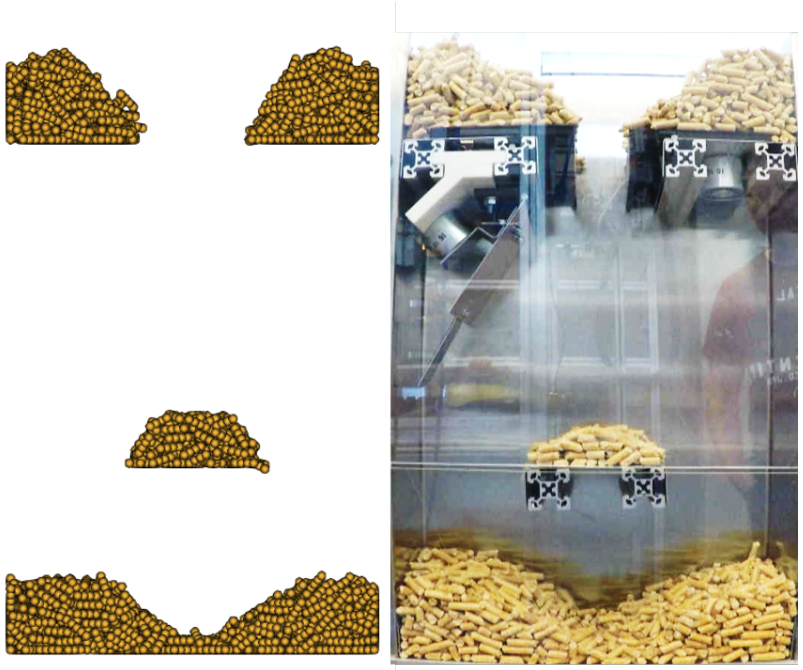

Figure 5.19: Comparison of the validation experiment with wood pellets (left: DEM simulation, right: Physical test in calibration unit) [211].

behavior, is shown in Figure 5.20. Within just 7 iterations the agent was able to find suitable input-parameters that match the target AoRs. Here, the actions strongly affect the AoRs and a decreasing of the step-sizes would further reduce the deviations . However, the validation test shows comparatively good results and approves visually the results of the DEM simulation as shown in Figure 5.21. The angle of repose as written in Table 5.7 confirms the correctness of the optimized parameters while the shear angle could not be measured accurately.

Table 5.7: Validation test results: Wet Sand.

	Shear Angle α [°]	**Angle of Repose** β [°]
Physical Test	-	66
Simulation Result	-	63.3

Summary

The novel DEM optimization procedure is the first attempt to use RL to obtain suitable DEM input parameters for desired materials. By using the developed automated mobile calibration unit materials can be calibrated at site. Moreover, the unit consists of all required calibration scenarios where the automation software guides through the different tests and assists the operator. Furthermore, the results and supplementary material are reliably saved into proper files. Afterwards, the developed optimization procedure is started and, depending on the desired

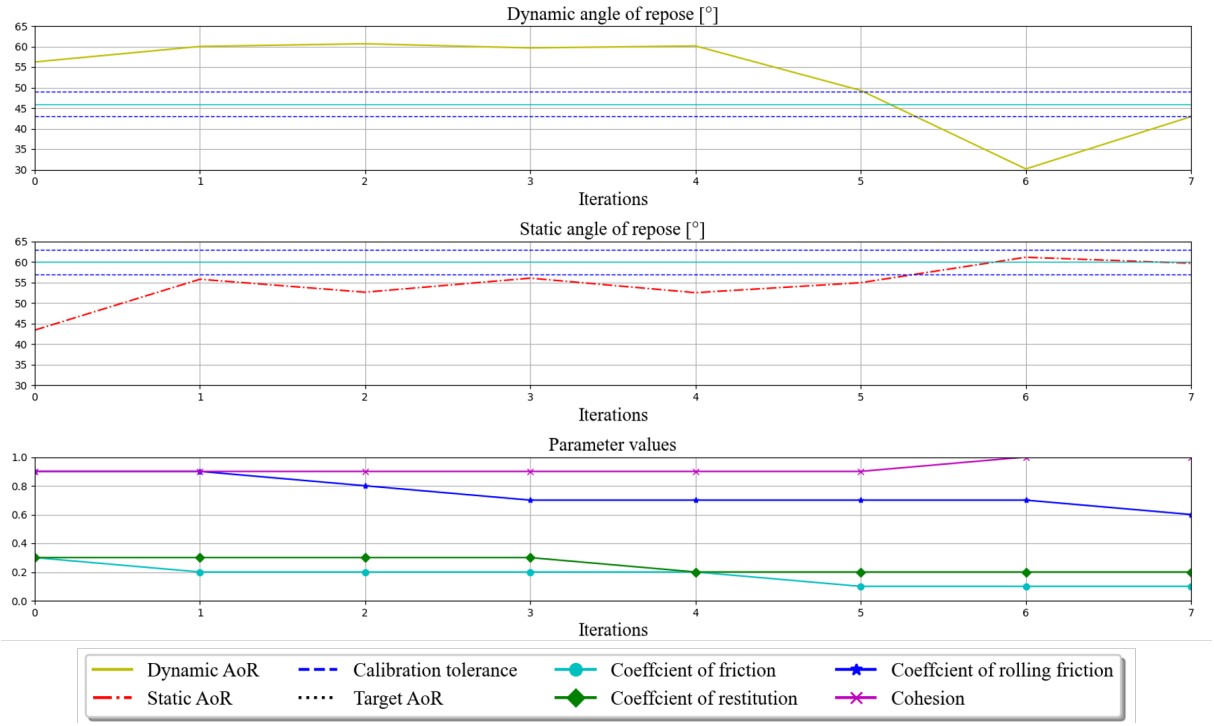

Figure 5.20: Calibration process of wet sand and results for the dynamic AoR (top), static AoR (middle) and parameter values [211].

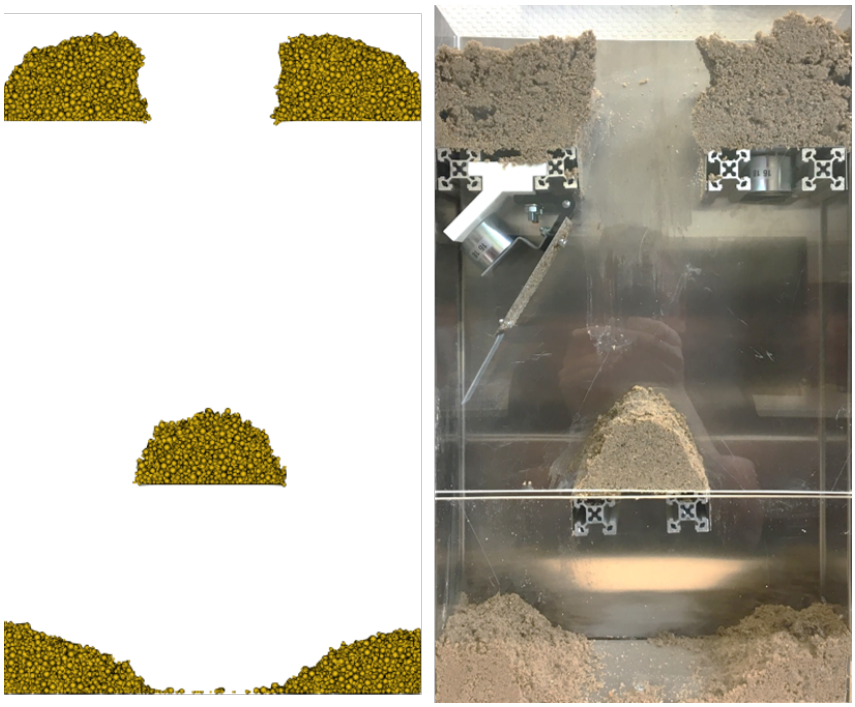

Figure 5.21: Comparison of the validation experiment with wet sand (left: DEM simulation, right: Physical test in calibration unit) [211].

material, the shape is approximated and the bulk density, etc. are measured. The developed

MORL-A2C, capable of combining at least two different objectives is then applied to train the agent with the pre-training data-set. This data-set, gathered by fully iterating through the state space, allows to fundamentally train the agent. After the sufficient pre-training with the proxy material and its intrinsic characteristics Θ_0, the agent switches to the DEM calibration by interacting with suitable digital twins of the physical tests and the desired material with Θ_x. Finally, the validation tests confirm the results and are used to evaluate the obtained parameters.

This novel procedure is able to find suitable DEM input parameters with the smallest number of required trials compared to the state-of-the-art approaches. In particular, the developed pre-training allows the agent to generalize and hence requires for the re-training only a few time-costly DEM simulations. The optimization of the three different materials yields to satisfying results and the approach contributes to remarkably improve future DEM optimizations.

6

Conclusion and Future Work

In this dissertation the coupling of RL with DEM-based digital twins and appropriate training methodologies are developed. The coupling and suitable methodologies are demonstrated on two examples, namely, the machine control of the PSM and the parameter optimization of DEM simulation input parameters and show overall remarkable results.

This chapter not only summarizes the entire work, methodologies and findings, but also gives an overview of possible subsequent future works.

6.1 Conclusion

To encounter the increasing complexity of modern machines and processes appropriate simulation models and suitable control algorithms are essential. Therefore, digital twins representing the physical counterpart in detail are used to develop sophisticated algorithms. Often digital twins are already available in the early stages of the product life-cycle and assist to develop algorithms, optimize the entire product and save commissioning time [121]. When it comes to control very complex processes, deep learning, and especially RL as a machine learning technique, has achieved remarkable results in many areas [86]. In order to manage machine control and parameter optimization tasks, this dissertation combines both, the digital twins and RL frameworks. The specialty of this work is the novel use of digital twins, which were modeled using the DEM simulation and are combined with RL. Therefore, this work discusses possible solutions for the defined main objectives, develops suitable methodologies and shows their applicability in multiple approaches.

After defining the state-of-the-art of the main subjects of the RL and the DEM, the required

basics are reflected in the theoretical basics, by explaining the feedforward neural networks, RL basic formulations and algorithms, the digital twin as well as the DEM computing, particle description and software. These fundamentals form the basis of the development of suitable methodologies to achieve convincing results in the following two applications.

The first application is the control of the newly developed PSM and in particular the control of two different parcel transport constellations. At fist the novel design and development of the entire PSM are presented. Derived from this design the digital twin of the PSM with the DEM is created. The PSM is conceptualized to transport and singulate parcels in modern logistic centers in the early stage of the sortation process. Using peristaltic waves as movement pattern, the PSM allows for gentle and versatile parcel transport. The DEM turns out to be an ideal tool to model the digital twin incorporating the dynamics of the mechanical parts and the trilateral interactions with the transport film and parcels. Due to the complex design of the PSM and interactions of the individual components a conventional control design is not very suitable and a machine learning approach is developed. However, the training of the RL agents from scratch while interacting with the DEM digital twin as a dynamic environment in order to achieve the control of the PSM solving transportation tasks with parcels, is hardly possible. The simulation time of the DEM models is too long to feasibly tune the required DNN structure, since the required training time would take weeks of computation. These drawbacks require special methodologies to deal with very complex and slow simulation models acting as environments in a RL control structure.

The first developed methodology describes the coupling of the DEM while wrapping it as a Python library which allows the use of the DEM in an object-oriented manner. By using Python, the DEM is extended to operate sequentially and event-based as well as to communicate with standard interfaces which are required for the further development of DEM models as dynamic environments. Additionally, Python inherently supports state-of-the-art machine learning libraries and therefore allows the integration of the DEM into RL frameworks. Two other methodologies namely the iterative learning schedule and the HRL approach are developed to reduce the complexity of high-dimensional control tasks and therefore reduce the entire training time of RL agents. Additionally, the novel methodology of the distributed ACRL learning allows to train on multiple DEM instances at the same time and speeds up the training supportively. All these methodologies allow to properly use RL in combination with DEM-based digital twins as environments for complex control tasks and parameter optimization. In order to validate the developed methods and to show the applicability and performance, they are used in two different control tasks namely the single- and multi-actuation transportation of parcels.

Applied to the first single-actuation transportation task, the iterative learning schedule with multi-complex environments and the parallel computation with the distributed ACRL learning, allow to successfully train the RL agent with the complex DEM environment in a reasonable time. By dividing the multi-actuation transportation task into different levels of the HRL framework,

the actuator-unit control and the orchestration of multiple units can be considered differently. In the HRL framework, the master-agent activates certain sub-agents or distinct sub-controls, so that the low-level sub-agents control the movement of the actuator-units individually. A proper pre-training in a simplified environment and a re-training for a few episodes with the DEM environment yield to a successful parcel transport. To conclusively shift the developed RL agents from the simulation environment to the real machine, a RL-PLC implementation is developed. This implementation allows the deployment and training of RL agents developed in Python, which are then directly used on Siemens IPCs and interact with real hardware. An exemplary adapted PSM task shows the applicability of pre-and re-training RL agents on the PLC.

The second application deals with the DEM parameter optimization and also requires especially developed methodologies. For valid DEM simulations of desired materials, it is mandatory to calibrate and subsequently optimize the DEM material input parameters. By considering the different targets which have to be fulfilled to reach a proper optimization, a special multi-objective RL approach is developed. Together with the developed pre-training strategy an optimization routine is developed which significantly reduces the required iteration of the entire optimization procedure. This procedure initially starts with the specially designed automated mobile calibration unit to calibrate and analyze the material characteristics. This unit, equipped with all relevant test procedures and supporting automation technology is used to analyze three different materials. While observing two important key parameters, namely the static and dynamic AoR the developed MORL-A2C agent learns to find suitable input parameters for different materials. The developed digital twins of the physical calibration tests are modeled and act as dynamic environments. Due to a pre-training that uses a pre-trained data-set, the agent is forced to generalize its learned knowledge to apply it on other materials. With a short re-training phase and only a few iterations the agent is then able to find optimal material parameters. The developed optimization procedure is able to analyze a broad range of different materials and shows remarkable performances and short computation times compared to other works.

This dissertation combines the two research fields of RL and the DEM. The work contributes methodologies for the coupling and handling of the training of RL agents with the computationally slow and high-dimensional DEM simulations. Therefore, the DEM-RL approach can be used to simulate and control or optimize complex problems and machines, even in areas which do not refer to bulk good handling. The developed methodologies are adaptable to other simulation methods and help to solve optimization tasks with high-dimensional and computationally slow environments.

6.2 Future Work

Since this work describes the basis for coupling RL and DEM approaches, the following deliberations based on these contributions are well worth considering. First of all, the development of the PSM could be continued in a way that the developed RL algorithms are tested on the real machine. Therefore, the trained agents of the single- and multi-actuation transportation task could be directly re-trained with the real machine and the developed RL-PLC implementation in a third phase. Therefore the iterative learning schedule could be enhanced to three stages with according update-rules to find proper DNN structures. The observations for both tasks can be obtained by the used camera system to detect the position of the parcel and take the positioning information of the used servo drives. Additionally, other tasks of the PSM like the singulation of bulky parcels or the transportation of multiple parcels could be focused.

By adjusting the step-size of the material parameter changes of the optimization procedure the accuracy of the developed approach could be increased. Furthermore, the use of e.g. the PPO instead of the A2C RL agent, could decrease the required training episodes and thus the entire training time. The pre-training data-set could be enhanced to a batch of different material types to reduce the variance between the pre-trained and researched materials. Using variants of pre-training data-sets could also simplify the degree of generalization of the agents and facilitate the pre-training.

Further, the developed combination of RL and the DEM can be used to optimize different kinds of machine types or processes. The DEM as powerful tool allows to simulate discrete and discontinuous materials, model complex movements of geometries, offer the possibility to approximate arbitrary shapes or forms bonds to generate a flexible behavior. In particular, in combination with Python, the applicability of the DEM is even more enhanced and can be used for any kind of data-driven approaches. Furthermore, the developed methodologies can be adapted to other simulation methods like the FEM or CFD and help to solve tasks in the fields of structural analysis or flow dynamics.

Bibliography

[1] R. Baheti and H. Gill, "Cyber-physical systems", *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

[2] Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-net: Learning PDEs from data", in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 3208–3216. [Online]. Available: https://proceedings.mlr.press/v80/long18a.html.

[3] L. Wright and S. Davidson, "How to tell the difference between a model and a digital twin", *Advanced Modeling and Simulation in Engineering Sciences*, vol. 7, no. 1, pp. 1–13, 2020.

[4] A. P. Markopoulos, N. E. Karkalos, and E.-L. Papazoglou, "Meshless methods for the simulation of machining and micro-machining: A review", *Archives of Computational Methods in Engineering*, vol. 27, no. 3, pp. 831–853, 2020.

[5] A. B. Yu, "Discrete element method", *Engineering Computations*, vol. 21, no. 2/3/4, pp. 205–214, 2004.

[6] S. Ji and S. Liang, "DEM-FEM-MBD coupling analysis of landing process of lunar lander considering landing mode and buffering mechanism", *Advances in Space Research*, 2021.

[7] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, *Agent57: Outperforming the atari human benchmark*, 2020.

[8] A. G. Barto, P. S. Thomas, and R. S. Sutton, "Some recent applications of reinforcement learning", in *Proceedings of the Eighteenth Yale Workshop on Adaptive and Learning Systems*, 2017.

[9] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play", *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow", *Trends in cognitive sciences*, vol. 23, no. 5, pp. 408–422, 2019.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning", *arXiv preprint arXiv:1312.5602*, 2013.

[13] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms", in *International Conference on Machine Learning*, 2014, pp. 387–395.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning", *arXiv preprint arXiv:1509.02971*, 2015.

[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning", in *International conference on machine learning*, 2016, pp. 1928–1937.

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization", in *International conference on machine learning*, 2015, pp. 1889–1897.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search", *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[20] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge", *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[21] Deepmind. (2020). Alphazero: Shedding new light on the grand games of chess, shogi and go, [Online]. Available: https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go (visited on 04/01/2022).

[22] Deepmind. (2020). Alphastar: Mastering the real-time strategy game StarCraft II, [Online]. Available: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii (visited on 04/01/2022).

[23] B. Chan, "OpenAI Five", *OpenAI*, vol. 2018, 2018.

[24] Rui Nian, Jinfeng Liu, and Biao Huang, "A review on reinforcement learning: Introduction and applications in industrial process control", *Computers & Chemical Engineering*, vol. 139, p. 106 886, 2020.

[25] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation", in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 590–595.

[26] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[27] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation", *arXiv preprint arXiv:1806.10293*, 2018.

[28] H. Fan, L. Zhu, C. Yao, J. Guo, and X. Lu, "Deep reinforcement learning for energy efficiency optimization in wireless networks", in *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, 2019, pp. 465–471.

[29] Y. Wang, K. Velswamy, and B. Huang, "A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems", *Processes*, vol. 5, no. 3, p. 46, 2017.

[30] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges", *Robotics*, vol. 8, no. 1, p. 4, 2019.

[31] Deepmind. (2020). Safety-first AI for autonomous data centre cooling and industrial control, [Online]. Available: https://deepmind.com/blog/article/safety-first-ai-autonomous-data-centre-cooling-and-industrial-control (visited on 04/01/2022).

[32] D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[33] S. X. Ding, *Advanced methods for fault diagnosis and fault-tolerant control*. Springer, 2021.

[34] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control", *IEEE circuits and systems magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[35] A. Heydari, "Stability analysis of optimal adaptive control using value iteration with approximation errors", *IEEE Transactions on Automatic Control*, vol. 63, no. 9, pp. 3119–3126, 2018.

[36] K. G. Vamvoudakis and F. L. Lewis, "Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem", *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[37] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming", *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012.

[38] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control", in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6023–6029.

[39] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, *Domain randomization for transferring deep neural networks from simulation to the real world*, 2017.

[40] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, *Sim-to-real: Learning agile locomotion for quadruped robots*, 2018.

[41] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey", in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 737–744.

[42] W. Chen, M. van Etten, T. Donohue, and K. Williams, "Application of the coupled discrete element modelling and modelica based multi-body dynamics in system-level modelling", in *International Conference on Discrete Element Methods*, 2016, pp. 571–578.

[43] S. Lommen, G. Lodewijks, and D. L. Schott, "Co-simulation framework of discrete element method and multibody dynamics models", *Engineering Computations*, vol. 35, no. 3, pp. 1481–1499, 2018.

[44] G. K. P. Barrios and L. M. Tavares, "A preliminary model of high pressure roll grinding using the discrete element method and multi-body dynamics coupling", *International Journal of Mineral Processing*, vol. 156, pp. 32–42, 2016.

[45] M. Michael, F. Vogel, and B. Peters, "DEM-FEM coupling simulations of the interactions between a tire tread and granular terrain", *Computer Methods in Applied Mechanics and Engineering*, vol. 289, pp. 227–248, 2015.

[46] N. Guo and J. Zhao, "A coupled FEM/DEM approach for hierarchical multiscale modelling of granular media", *International Journal for Numerical Methods in Engineering*, vol. 99, no. 11, pp. 789–818, 2014.

[47] T. Tsuji, K. Yabumoto, and T. Tanaka, "Spontaneous structures in three-dimensional bubbling gas-fluidized bed by parallel DEM-CFD coupling simulation", *Powder Technology*, vol. 184, no. 2, pp. 132–140, 2008.

[48] C. Goniva, C. Kloss, A. Hager, and S. Pirker, "An open source CFD-DEM perspective", in *Proceedings of OpenFOAM Workshop, Göteborg*, 2010, pp. 22–24.

[49] D. Liu, C. Bu, and X. Chen, "Development and test of CFD-DEM model for complex geometry: A coupling algorithm for fluent and dem", *Computers & Chemical Engineering*, vol. 58, pp. 260–268, 2013.

[50] A. Alexiadis, M. J. Simmons, K. Stamatopoulos, H. K. Batchelor, and I. Moulitsas, "The duality between particle methods and artificial neural networks", *Scientific RepoRtS*, vol. 10, no. 1, pp. 1–7, 2020.

[51] F. Westbrink and A. Schwung, "Virtual commissioning approach based on the discrete element method", in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, IEEE, 2018, pp. 424–429.

[52] Adam Kešner, Rostislav Chotěborský, Miloslav Linda, Monika Hromasová, Egidijus Katinas, and Hadi Sutanto, "Stress distribution on a soil tillage machine frame segment with a chisel shank simulated using discrete element and finite element methods and validate by experiment", *Biosystems Engineering*, vol. 209, pp. 125–138, 2021.

[53] A. Munjiza, H. Smoljanović, N. Živaljić, A. Mihanovic, V. Divić, I. Uzelac, Ž. Nikolić, I. Balić, and B. Trogrlić, "Structural applications of the combined finite–discrete element method", *Computational Particle Mechanics*, vol. 7, no. 5, pp. 1029–1046, 2020.

[54] W. Song, B. Huang, X. Shu, J. Stránský, and H. Wu, "Interaction between railroad ballast and sleeper: A DEM-FEM approach", *International Journal of Geomechanics*, vol. 19, no. 5, p. 04 019 030, 2019.

[55] C. Richter, “Gekoppelte Diskrete Elemente und Mehrkörpersimulation am Beispiel von Becherförderern”, PhD thesis, Otto-von-Guericke-Universität Magdeburg, Fakultät für Maschinenbau, 2020.

[56] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauss, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, *et al.*, “The functional mockup interface for tool independent exchange of simulation models”, in *Proceedings of the 8th International Modelica Conference*, 2011, pp. 105–114.

[57] Christian Richter, Thomas Roessler, Hendrik Otto, and André Katterfeld, “Coupled discrete element and multibody simulation, part i implementation, verification and validation”, *Powder Technology*, 2020.

[58] EDEM Simulation. (2019). Generic EDEM material model (GEMM) database, [Online]. Available: https://community.altair.com/community?id=community_Blog&sys_id=cce0331cdbef2410e8863978f49619e4 (visited on 04/01/2022).

[59] C. M. Wensrich and A. Katterfeld, “Rolling friction as a technique for modelling particle shape in DEM”, *Powder Technology*, vol. 217, pp. 409–417, 2012.

[60] S. C. Thakur, J. P. Morrissey, J. Sun, J. F. Chen, and J. Y. Ooi, “Micromechanical analysis of cohesive granular materials using the discrete element method with an adhesive elasto-plastic contact model”, *Granular Matter*, vol. 16, no. 3, pp. 383–400, 2014.

[61] K. Tanaka, M. Nishida, T. Kunimochi, and T. Takagi, “Discrete element simulation and experiment for dynamic response of two-dimensional granular matter to the impact of a spherical projectile”, *Powder Technology*, vol. 124, no. 1-2, pp. 160–173, 2002.

[62] Asaf, Z and Rubinstein, D and Shmulevich, I, “Determination of discrete element model parameters required for soil tillage”, *Soil and Tillage Research*, vol. 92, no. 1-2, pp. 227–242, 2007.

[63] C. J. Coetzee, “Review: Calibration of the discrete element method”, *Powder Technology*, vol. 310, pp. 104–142, 2017.

[64] T. Roessler, C. Richter, A. Katterfeld, and F. Will, “Development of a standard calibration procedure for the DEM parameters of cohesionless bulk materials–part I: Solving the problem of ambiguous parameter combinations”, *Powder Technology*, vol. 343, pp. 803–812, 2019.

[65] B. El Kassem, N. Salloum, T. Brinz, Y. Heider, and B. Markert, “A semi-automated DEM parameter calibration technique of powders based on different bulk responses extracted from auger dosing experiments”, *KONA Powder and Particle Journal*, 2020.

[66] Y. C. Zhou, B. H. Xu, A.-B. Yu, and P. Zulli, "An experimental and numerical study of the angle of repose of coarse spheres", *Powder Technology*, vol. 125, no. 1, pp. 45–54, 2002.

[67] H. M. B. Al-Hashemi and O. S. B. Al-Amoudi, "A review on the angle of repose of granular materials", *Powder Technology*, vol. 330, pp. 397–417, 2018.

[68] T. Roessler and A. Katterfeld, "DEM parameter calibration of cohesive bulk materials using a simple angle of repose test", *Particuology*, vol. 45, pp. 105–115, 2019.

[69] G. Lumay, N. M. Tripathi, P. Scientist, and F. Francqui, "How to gain a full understanding of powder flow properties, and the benefits of doing so", *ONdrugDelivery Mag*, vol. 102, pp. 42–46, 2019.

[70] H. N. Pitanga, J.-P. Gourc, and O. M. Vilar, "Interface shear strength of geosynthetics: Evaluation and analysis of inclined plane tests", *Geotextiles and Geomembranes*, vol. 27, no. 6, pp. 435–446, 2009.

[71] C. J. Coetzee, "Calibration of the discrete element method and the effect of particle shape", *Powder Technology*, vol. 297, pp. 50–70, 2016.

[72] Z. Chen, C. Wassgren, E. Veikle, and K. Ambrose, "Determination of material and interaction properties of maize and wheat kernels for DEM simulation", *Biosystems Engineering*, vol. 195, pp. 208–226, 2020.

[73] D. Höhner, S. Wirtz, and V. Scherer, "Experimental and numerical investigation on the influence of particle shape and shape approximation on hopper discharge using the discrete element method", *Powder Technology*, vol. 235, pp. 614–627, 2013.

[74] D. Schulze, *Powders and bulk solids: Behavior, characterization, storage and flow*. Berlin: Springer, 2008.

[75] A. V. Boikov, R. V. Savelev, and V. A. Payor, "DEM calibration approach: Design of experiment", in *Journal of Physics: Conference Series*, vol. 1015, 2018, p. 032 017.

[76] S. Zhang, M. Z. Tekeste, Y. Li, A. Gaul, D. Zhu, and J. Liao, "Scaled-up rice grain modelling for DEM calibration and the validation of hopper flow", *Biosystems Engineering*, vol. 194, pp. 196–212, 2020.

[77] S. B. Turkia, D. N. Wilke, P. Pizette, N. Govender, and N.-E. Abriak, "Benefits of virtual calibration for discrete element parameter estimation from bulk experiments", *Granular Matter*, vol. 21, no. 4, p. 110, 2019.

[78] H. Q. Do, A. M. Aragón, and D. L. Schott, "A calibration framework for discrete element model parameters using genetic algorithms", *Advanced Powder Technology*, vol. 29, no. 6, pp. 1393–1403, 2018.

[79] B. El-Kassem, N. Salloum, T. Brinz, Y. Heider, and B. Markert, "A multivariate regression parametric study on DEM input parameters of free-flowing and cohesive powders with experimental data-based validation", *Computational Particle Mechanics*, pp. 1–25, 2020.

[80] C. Richter, T. Rößler, G. Kunze, A. Katterfeld, and F. Will, "Development of a standard calibration procedure for the DEM parameters of cohesionless bulk materials–part II: Efficient optimization-based calibration", *Powder Technology*, vol. 360, pp. 967–976, 2020.

[81] F. Ye, C. Wheeler, B. Chen, J. Hu, K. Chen, and W. Chen, "Calibration and verification of DEM parameters for dynamic particle flow conditions using a backpropagation neural network", *Advanced Powder Technology*, vol. 30, no. 2, pp. 292–301, 2019.

[82] L. Benvenuti, C. Kloss, and S. Pirker, "Identification of DEM simulation parameters by artificial neural networks and bulk experiments", *Powder Technology*, vol. 291, pp. 456–465, 2016.

[83] M. Alnaggar and N. Bhanot, "A machine learning approach for the identification of the lattice discrete particle model parameters", *Engineering Fracture Mechanics*, vol. 197, pp. 160–175, 2018.

[84] R. Kumar, C. M. Patel, A. K. Jana, and S. R. Gopireddy, "Prediction of hopper discharge rate using combined discrete element method and artificial neural network", *Advanced Powder Technology*, vol. 29, no. 11, pp. 2822–2834, 2018.

[85] Z. Hu, X. Liu, and C. Chu, "DEM data-driven modeling of repose angle of granular materials", in *2020 2nd International Conference on Industrial Artificial Intelligence (IAI)*, 2020, pp. 1–6.

[86] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[87] K. Hornik, "Approximation capabilities of multilayer feedforward networks", *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[88] E. Charniak, *Introduction to Deep Learning*. The MIT Press, 2019.

[89] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018.

[90] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 2.

[91] J. R. Norris, *Markov chains*. Cambridge University Press, 1998.

[92] D. Silver, "Reinforcement learning and simulation-based search in computer go", PhD thesis, University of Alberta Libraries, 2009.

[93] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning”, *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[94] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, “Natural actor–critic algorithms”, *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[95] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, “Sample efficient actor-critic with experience replay”, *arXiv preprint arXiv:1611.01224*, 2016.

[96] K. van Moffaert and A. Nowé, “Multi-objective reinforcement learning using sets of pareto dominating policies”, *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.

[97] K. van Moffaert, M. M. Drugan, and A. Nowé, “Scalarized multi-objective reinforcement learning: Novel design techniques”, in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2013, pp. 191–199.

[98] C. Liu, X. Xu, and D. Hu, “Multiobjective reinforcement learning: A comprehensive overview”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2015.

[99] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, “Empirical evaluation methods for multiobjective reinforcement learning algorithms”, *Machine Learning*, vol. 84, no. 1-2, pp. 51–80, 2011.

[100] H. Iima and Y. Kuroe, “Multi-objective reinforcement learning for acquiring all pareto optimal policies simultaneously-method of determining scalarization weights”, in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2014, pp. 876–881.

[101] R. Yang, X. Sun, and K. Narasimhan, “A generalized algorithm for multi-objective reinforcement learning and policy adaptation”, in *Advances in Neural Information Processing Systems*, 2019, pp. 14 636–14 647.

[102] H. Dong, Z. Ding, and S. Zhang, *Deep reinforcement learning: Fundamentals, research and applications*. Singapore: Springer, 2020.

[103] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, “Hierarchical reinforcement learning: A comprehensive survey”, *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.

[104] M. Baykal-Gürsoy, “Semi-markov decision processes”, *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[105] Richard S. Sutton, Doina Precup, and Satinder Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning", *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[106] P. Dayan and G. E. Hinton, *Feudal reinforcement learning. nips'93 (pp. 271–278)*, 1993.

[107] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition", *Journal of artificial intelligence research*, vol. 13, pp. 227–303, 2000.

[108] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines", *Advances in neural information processing systems*, vol. 10, pp. 1043–1049, 1997.

[109] P.-L. Bacon, J. Harb, and D. Precup, *The option-critic architecture*, 2016.

[110] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, *FeUdal networks for hierarchical reinforcement learning*, 2017.

[111] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Modeling, simulation, information technology & processing roadmap", *National Aeronautics and Space Administration*, 2012.

[112] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems", in *Transdisciplinary perspectives on complex systems*, Springer, 2017, pp. 85–113.

[113] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn, "Digital twin in manufacturing: A categorical literature review and classification", *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

[114] A. Madni, C. Madni, and S. Lucero, "Leveraging digital twin technology in model-based systems engineering", *Systems*, vol. 7, no. 1, p. 7, 2019.

[115] S. Boschert and R. Rosen, "Digital twin—the simulation aspect", in *Mechatronic futures*, Springer, 2016, pp. 59–74.

[116] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann, "Experimentable digital twins—streamlining simulation-based systems engineering for industry 4.0", *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1722–1731, 2018.

[117] M. Schluse, L. Atorf, and J. Rossmann, "Experimentable digital twins for model-based systems engineering and simulation-based development", in *2017 Annual IEEE International Systems Conference (SysCon)*, 2017, pp. 1–8.

[118] F. Jaensch, A. Csiszar, C. Scheifele, and A. Verl, "Digital twins of manufacturing systems as a base for machine learning", in *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2018, pp. 1–6.

[119] I. Verner, D. Cuperman, A. Fang, M. Reitman, T. Romm, and G. Balikin, "Robot online learning through digital twin experiments: A weightlifting project", in *Online Engineering & Internet of Things*, M. E. Auer and D. G. Zutin, Eds., ser. Lecture Notes in Networks and Systems, Springer International Publishing, 2017, pp. 307–314.

[120] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for UAV attitude control", *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[121] Kaishu Xia, Christopher Sacco, Max Kirkpatrick, Clint Saidy, Lam Nguyen, Anil Kircaliali, and Ramy Harik, "A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence", *Journal of Manufacturing Systems*, 2020.

[122] M. Fritz, A. Wolfschluckner, and D. Jodin, "Simulation von Paketen im Pulk", *Logistics Journal*, vol. 2013, no. 11, 2013.

[123] P. A. Cundall, "A computer model for simulating progressive, large-scale movement in blocky rock system", *Proceedings of the International Symposium on Rock Mechanics*, pp. 129–136, 1971.

[124] P. A. Cundall and O. D. L. Strack, "A discrete numerical model for granular assemblies", *geotechnique*, vol. 29, no. 1, pp. 47–65, 1979.

[125] H. P. Zhu, Z. Y. Zhou, R. Y. Yang, and A. B. Yu, "Discrete particle simulation of particulate systems: A review of major applications and findings", *Chemical engineering science*, vol. 63, no. 23, pp. 5728–5770, 2008.

[126] L. Jing and O. Stephansson, *Fundamentals of discrete element methods for rock engineering: Theory and applications*. Elsevier, 2007, vol. 85.

[127] R. Bharadwaj *et al.*, "Using DEM to solve bulk material handling problems", *Chemical Engineering Progress*, vol. 108, no. 9, pp. 54–58, 2012.

[128] D. Ilic, A. Roberts, and C. Wheeler, "Modelling bulk solid interactions in transfer chutes: Accelerated flow", *Chemical engineering science*, vol. 209, p. 115 197, 2019.

[129] D. Wei, R. C. Hurley, L. H. Poh, D. Dias-da-Costa, and Y. Gan, "The role of particle morphology on concrete fracture behaviour: A meso-scale modelling approach", *Cement and Concrete Research*, vol. 134, p. 106 096, 2020.

[130] F. Fleissner, T. Gaugele, and P. Eberhard, "Applications of the discrete element method in mechanical engineering", *Multibody system dynamics*, vol. 18, no. 1, p. 81, 2007.

[131] H. Nakashima, H. Fujii, A. Oida, M. Momozu, H. Kanamori, S. Aoki, T. Yokoyama, H. Shimizu, J. Miyasaka, and K. Ohdoi, "Discrete element method analysis of single wheel performance for a small lunar rover on sloped terrain", *Journal of Terramechanics*, vol. 47, no. 5, pp. 307–321, 2010.

[132] Y. Li, W. Wu, X. Chu, and W. Zou, "Effects of stress paths on triaxial compression mechanical properties of QH-E lunar soil simulant studied by DEM simulation", *Granular Matter*, vol. 22, no. 2, pp. 1–10, 2020.

[133] D. Prims, J. Kötz, S. Göttlich, and A. Katterfeld, "Validation of flow models as new simulation approach for parcel handling in bulk mode", *arXiv preprint arXiv:1901.08482*, 2019.

[134] G. S. Chadha, F. Westbrink, T. Schütte, and A. Schwung, "Optimal dosing of bulk material using mass-flow estimation and DEM simulation", in *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 256–261.

[135] J. Horabik, P. Parafiniuk, and M. Molenda, "Stress profile in bulk of seeds in a shallow model silo as influenced by mobilisation of particle-particle and particle-wall friction: Experiments and DEM simulations", *Powder Technology*, vol. 327, pp. 320–334, 2018.

[136] P. W. Cleary, "Predicting charge motion, power draw, segregation and wear in ball mills using discrete element methods", *Minerals Engineering*, vol. 11, no. 11, pp. 1061–1080, 1998.

[137] S. Lobo-Guerrero and L. E. Vallejo, "Discrete element method analysis of railtrack ballast degradation during cyclic loading", *Granular Matter*, vol. 8, no. 3-4, p. 195, 2006.

[138] Y. P. Cheng, Y. Nakata, and M. D. Bolton, "Discrete element simulation of crushable soil", *geotechnique*, vol. 53, no. 7, pp. 633–641, 2003.

[139] Itasca Consulting Group, Inc. (2018) PFC, *Particle flow code, ver. 6.0. minneapolis: Itasca*.

[140] M. Otsubo, C. O'Sullivan, and T. Shire, "Empirical assessment of the critical time increment in explicit particulate discrete element method simulations", *Computers and Geotechnics*, vol. 86, pp. 67–79, 2017.

[141] D. Rathbone, M. Marigo, D. Dini, and B. van Wachem, "An accurate force–displacement law for the modelling of elastic–plastic contacts in discrete element simulations", *Powder Technology*, vol. 282, pp. 2–9, 2015.

[142] C. Kloss, C. Goniva, A. Hager, S. Amberger, and S. Pirker, "Models, algorithms and validation for opensource dem and CFD-DEM", *Progress in Computational Fluid Dynamics, an International Journal*, vol. 12, no. 2-3, pp. 140–152, 2012.

[143] H. Kruggel-Emden, E. Simsek, S. Rickelt, S. Wirtz, and V. Scherer, "Review and extension of normal force models for the discrete element method", *Powder Technology*, vol. 171, no. 3, pp. 157–173, 2007.

[144] B. Soltanbeigi, A. Podlozhnyuk, S.-A. Papanicolopulos, C. Kloss, S. Pirker, and J. Y. Ooi, "DEM study of mechanical characteristics of multi-spherical and superquadric particles at micro and macro scales", *Powder Technology*, vol. 329, pp. 288–303, 2018.

[145] H. Kruggel-Emden, S. Rickelt, S. Wirtz, and V. Scherer, "A study on the validity of the multi-sphere discrete element method", *Powder Technology*, vol. 188, no. 2, pp. 153–165, 2008.

[146] M. Price, V. Murariu, and G. Morrison, "Sphere clump generation and trajectory comparison for real particles", *Proceedings of Discrete Element Modelling 2007*, 2007.

[147] F. Westbrink and A. Schwung, "Improved approximation of arbitrary shapes in DEM simulations with multi-spheres", in *COMPLAS XIV: proceedings of the XIV International Conference on Computational Plasticity: fundamentals and applications*, 2017, pp. 854–865.

[148] A. Podlozhnyuk, S. Pirker, and C. Kloss, "Efficient implementation of superquadric particles in discrete element method within an open-source framework", *Computational Particle Mechanics*, vol. 4, no. 1, pp. 101–118, 2017.

[149] A. H. Barr, "Superquadrics and angle-preserving transformations", *IEEE Computer graphics and Applications*, vol. 1, no. 1, pp. 11–23, 1981.

[150] D. O. Potyondy and P. A. Cundall, "A bonded-particle model for rock", *International journal of rock mechanics and mining sciences*, vol. 41, no. 8, pp. 1329–1364, 2004.

[151] T. T. Nguyen and B. Indraratna, "Hydraulic behaviour of parallel fibres under longitudinal flow: A numerical treatment", *Canadian Geotechnical Journal*, vol. 53, no. 7, pp. 1081–1092, 2016.

[152] M. Schramm, M. Z. Tekeste, C. Plouffe, and D. Harby, "Estimating bond damping and bond young's modulus for a flexible wheat straw discrete element method model", *Biosystems Engineering*, vol. 186, pp. 349–355, 2019.

[153] Y. Guo, J. Curtis, C. Wassgren, W. Ketterhagen, and B. Hancock, "Granular shear flows of flexible rod-like particles", in *AIP Conference Proceedings*, vol. 1542, 2013, pp. 491–494.

[154] Y. Guo, C. Wassgren, B. Hancock, W. Ketterhagen, and J. Curtis, "Validation and time step determination of discrete element modeling of flexible fibers", *Powder Technology*, vol. 249, pp. 386–395, 2013.

[155] A. Katterfeld and C. Wensrich, "Understanding granular media: From fundamentals and simulations to industrial application", *Granular Matter*, vol. 19, no. 4, p. 83, 2017.

[156] F. P. André, A. Potapov, C. Maliska Jr, and L. M. Tavares, "Simulation of single particle breakage using non-round particles in Rocky DEM", in *26th Int. Min. Congr. Exhib. Turkey, Antalya*, vol. 1, 2019, pp. 981–990.

[157] V. Smilauer, E. Catalano, B. Chareyre, S. Dorofeenko, J. Duriez, N. Dyck, J. Elias, B. Er, A. Eulitz, A. Gladky, N. Guo, C. Jakob, F. Kneib, J. Kozicki, D. Marzougui, R. Maurin, C. Modenese, L. Scholtes, L. Sibille, J. Stransky, T. Sweijen, K. Thoeni, and C. Yuan, *Yade Documentation 2Nd Ed*. Zenodo, 2015.

[158] C. Kloss and C. Goniva, "LIGGGHTS – open source discrete element simulations of granular materials based on Lammps", in *Supplemental Proceedings*, John Wiley & Sons, Ltd, 2011, pp. 781–788.

[159] R. Berger, C. Kloss, A. Kohlmeyer, and S. Pirker, "Hybrid parallelization of the LIGGGHTS open-source DEM code", *Powder Technology*, vol. 278, pp. 234–247, 2015.

[160] J. Ahrens, B. Geveci, and C. Law, "Paraview: An end-user tool for large data visualization", *The visualization handbook*, vol. 717, 2005.

[161] E. Zákorová, "E-commerce and its impact on logistics requirements", *Open Engineering*, vol. 7, no. 1, pp. 121–125, 2017.

[162] T. Brown. (2019). Choosing a singulation method, [Online]. Available: https://parcelindustry.com/article-4901-Choosing-a-Singulation-Method.html (visited on 04/01/2022).

[163] Beumer Group. (2019). Automatisierte Paket-Vereinzelung - BEUMER Group, [Online]. Available: https://www.beumergroup.com/de/i/automatic-parcel-singulator-23674/ (visited on 04/01/2022).

[164] Siemens. (2018). Siemens offers innovative products for efficient singulation and splitting of parcel, [Online]. Available: https://www.siemens-logistics.com/en/news/press-releases/siemens-offers-innovative-products-for-efficient-singulation-and-splitting-of-parcel-flows.

[165] Fives Group. (2019). Accord singulator, [Online]. Available: https://www.fivesgroup.com/smart-automation-solutions/technologies/singulators/the-accord-packet-singulator (visited on 04/01/2022).

[166] C. Pozrikidis, "A study of peristaltic flow", *Journal of Fluid Mechanics*, vol. 180, pp. 515–527, 1987.

[167] M. Y. Jaffrin and A. H. Shapiro, "Peristaltic pumping", *Annual Review of Fluid Mechanics*, vol. 3, no. 1, pp. 13–37, 1971.

[168] J. Klespitz and L. Kovács, "Peristaltic pumps — a review on working and control possibilities", in *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2014, pp. 191–194.

[169] Y. C. Fung and C. S. Yih, "Peristaltic transport", *Journal of Applied Mechanics*, vol. 35, no. 4, pp. 669–675, 1968.

[170] N. Saga and T. Nakamura, "Development of a peristaltic crawling robot using magnetic fluid on the basis of the locomotion mechanism of the earthworm", *Smart materials and structures*, vol. 13, no. 3, p. 566, 2004.

[171] Y. Mano, R. Ishikawa, Y. Yamada, and T. Nakamura, "Development of high-speed type peristaltic crawling robot for long-distance and complex-line sewer pipe inspection", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 8177–8183.

[172] Z. Deng, M. Stommel, and W. Xu, "Pneumatic system and low-level control of a soft machine table inspired by caterpillar locomotion", in *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC)*, 2016, pp. 364–369.

[173] Z. Deng, M. Stommel, and W. Xu, "Mechatronics design, modeling, and characterization of a soft robotic table for object manipulation on surface", *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 6, pp. 2715–2725, 2018.

[174] M. Stommel and W. Xu, "Learnability of the moving surface profiles of a soft robotic sorting table", *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1581–1587, 2016.

[175] M. Stommel and W. Xu, "Optimal, efficient sequential control of a soft-bodied, peristaltic sorting table", *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 858–867, 2016.

[176] R. Hashem, B. Smith, D. Browne, W. Xu, and M. Stommel, "Control of a soft-bodied xy peristaltic table for delicate sorting", in *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC)*, IEEE, 2016, pp. 358–363.

[177] R. M. McKenzie, J. O. Roberts, M. E. Sayed, and A. A. Stokes, "Perisim: A simulator for optimizing peristaltic table control", *Advanced Intelligent Systems*, vol. 1, no. 8, p. 1 900 070, 2019.

[178] Festo. (2014). Wavehandling | festo corporate, [Online]. Available: https://www.festo.com/group/en/cms/10225.htm (visited on 04/01/2022).

[179] W. Lee, N. Lee, J.-W. Kim, M. Shin, and J. Lee, "MoleBot': An organic user-interface-based robot that provides users with richer kinetic interactions", *Interacting with Computers*, vol. 25, no. 2, pp. 154–172, 2013.

[180] S. Follmer, D. Leithinger, A. Olwal, A. Hogge, and H. Ishii, "InFORM: Dynamic physical affordances and constraints through shape and object actuation", in *Uist*, vol. 13, 2013, pp. 2 501 988–2 502 032.

[181] K. Schenk and J. Lunze, "Fault tolerance in networked systems through flexible task assignment", in *2019 4th Conference on Control and Fault Tolerant Systems (SysTol)*, 2019, pp. 257–263.

[182] L. Cao, K. Richter, C. Richter, and A. Katterfeld, *Simulation der peristaltischen Förderung von Stückgütern als Schüttgut*. Wissenschaftliche Gesellschaft für Technische Logistik, 2014.

[183] F. Westbrink, R. Sivanandan, T. Schütte, and A. Schwung, "Design approach and simulation of a peristaltic sortation machine", in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2019, pp. 1127–1132.

[184] F. Westbrink, A. Schwung, and S. X. Ding, "How to get a parcel surfing", in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 1619–1624.

[185] F. Westbrink, A. Schwung, and S. X. Ding, "Data-based control of peristaltic sortation machines using discrete element method", in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 575–580.

[186] N. Ketkar, "Introduction to pytorch", in *Deep learning with python*, Springer, 2017, pp. 195–208.

[187] Luc Scholtès and Frédéric-Victor Donzé, "A DEM model for soft and hard rocks: Role of grain interlocking on strength", *Journal of the Mechanics and Physics of Solids*, vol. 61, no. 2, pp. 352–369, 2013.

[188] S. Raschka, *Python machine learning*. Packt publishing ltd, 2015.

[189] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with CUDA", *IEEE Micro*, vol. 28, no. 4, pp. 13–27, 2008.

[190] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library", in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[191] A. Abi-Mansour, "Pygran: An object-oriented library for DEM simulation and analysis", *SoftwareX*, vol. 9, pp. 168–174, 2019.

[192] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning", in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[193] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey", *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1–50, 2020.

[194] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, *What matters in on-policy reinforcement learning? a large-scale empirical study*, 2020.

[195] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014.

[196] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[197] H. Z. Alemu, W. Wu, and J. Zhao, "Feedforward neural networks with a hidden layer regularization method", *Symmetry*, vol. 10, no. 10, p. 525, 2018.

[198] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay", *arXiv preprint arXiv:1803.00933*, 2018.

[199] H. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and Daumé, III, Hal, "Hierarchical imitation and reinforcement learning", in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 2917–2926.

[200] B. Hengst, "Hierarchical reinforcement learning", in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Springer US, 2010, pp. 495–502.

[201] I. P. Durugkar, C. Rosenbaum, S. Dernbach, and S. Mahadevan, "Deep reinforcement learning with macro-actions", *arXiv preprint arXiv:1606.04615*, 2016.

[202] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. L. Dean, and C. Boutilier, "Hierarchical solution of markov decision processes using macro-actions", *arXiv preprint arXiv:1301.7381*, 2013.

[203] Y. Flet-Berliac and P. Preux, *MERL: Multi-head reinforcement learning*, 2019.

[204] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry, "Implementation matters in deep RL: A case study on PPO and TRPO", in *International Conference on Learning Representations*, 2020.

[205] R. Langmann and L. F. Rojas-Peña, “A PLC as an industry 4.0 component”, in *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, 2016, pp. 10–15.

[206] Alberto Diez-Olivan, Javier Del Ser, Diego Galar, and Basilio Sierra, “Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0”, *Information Fusion*, vol. 50, pp. 92–111, 2019.

[207] F. Westbrink, G. S. Chadha, and A. Schwung, “Integrated IPC for data-driven fault detection”, in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 277–282.

[208] Siemens Global Website. (2020). Simatic s7-1500 tm npu | simatic s7-1500 | siemens global, [Online]. Available: https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1500/simatic-s7-1500-tm-npu.html (visited on 04/01/2022).

[209] Beckhoff. (2020). Beckhoff new automation technology, [Online]. Available: https://www.beckhoff.com/de-de/produkte/automation/twincat/tfxxxx-twincat-3-functions/tf3xxx-tc3-measurement/tf3810.html (visited on 04/01/2022).

[210] H. Berger, *Automatisieren mit SIMATIC: Hardware und Software, Projektierung und Programmierung, Datenkommunikation, Bedienen und Beobachten*, 6. Aufl. s.l.: Publicis, 2016.

[211] F. Westbrink, A. Elbel, A. Schwung, and S. X. Ding, “Optimization of DEM parameters using multi-objective reinforcement learning”, *Powder Technology*, vol. 379, pp. 602–616, 2021.

[212] F. Westbrink, A. Elbel, and A. Schwung, “Development of an automated mobile DEM calibration unit”, *Proceedings of the 8th International Conference on Discrete Element Methods*, 2019.

[213] J. M. Tiscar, A. Escrig, G. Mallol, J. Boix, and F. A. Gilabert, “DEM-based modelling framework for spray-dried powders in ceramic tiles industry. part I: Calibration procedure”, *Powder Technology*, vol. 356, pp. 818–831, 2019.

[214] C. J. Coetzee, “Particle upscaling: Calibration and validation of the discrete element method”, *Powder Technology*, vol. 344, pp. 487–503, 2019.

[215] S. Lommen, M. Mohajeri, G. Lodewijks, and D. Schott, “DEM particle upscaling for large-scale bulk handling equipment and material interaction”, *Powder Technology*, vol. 352, pp. 273–282, 2019.

[216] T. Roessler and A. Katterfeld, “Scaling of the angle of repose test and its influence on the calibration of DEM parameters using upscaled particles”, *Powder Technology*, vol. 330, pp. 58–66, 2018.

[217] J. Grobbel, S. Brendelberger, M. Henninger, C. Sattler, and R. Pitz-Paal, “Calibration of parameters for DEM simulations of solar particle receivers by bulk experiments and surrogate functions”, *Powder Technology*, vol. 364, pp. 831–844, 2020.

[218] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, “A study on overfitting in deep reinforcement learning”, *arXiv preprint arXiv:1804.06893*, 2018.

[219] A. Zhang, N. Ballas, and J. Pineau, “A dissection of overfitting and generalization in continuous reinforcement learning”, *arXiv preprint arXiv:1806.07937*, 2018.

List of Publications

Journal Paper

1. F. Westbrink, A. Elbel, A. Schwung, *et al.*, “Optimization of DEM parameters using multi-objective reinforcement learning”, *Powder Technology*, vol. 379, pp. 602–616, 2021

Conference Papers

1. F. Westbrink, A. Schwung, and S. X. Ding, “Data-based control of peristaltic sortation machines using discrete element method”, in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 575–580

2. F. Westbrink, A. Schwung, and S. X. Ding, “How to get a parcel surfing”, in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 1619–1624

3. F. Westbrink, A. Elbel, and A. Schwung, “Development of an automated mobile DEM calibration unit”, *Proceedings of the 8th International Conference on Discrete Element Methods*, 2019

4. F. Westbrink, R. Sivanandan, T. Schütte, *et al.*, “Design approach and simulation of a peristaltic sortation machine”, in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2019, pp. 1127–1132

5. F. Westbrink and A. Schwung, “Virtual commissioning approach based on the discrete element method”, in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, IEEE, 2018, pp. 424–429

6. F. Westbrink, G. S. Chadha, and A. Schwung, “Integrated IPC for data-driven fault detection”, in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 277–282

7. G. S. Chadha, F. Westbrink, T. Schütte, *et al.*, “Optimal dosing of bulk material using mass-flow estimation and DEM simulation”, in *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 256–261

8. F. Westbrink and A. Schwung, “Improved approximation of arbitrary shapes in DEM simulations with multi-spheres”, in *COMPLAS XIV: proceedings of the XIV International Conference on Computational Plasticity: fundamentals and applications*, 2017, pp. 854–865