NAWI Graz
Natural Sciences

Gerda Langer, B.Sc.

Physics-Informed Neural Networks for Applications in Quantum Chemistry and Thermodynamics

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Physics

submitted to

Graz University of Technology

Supervisor

Assoc. Prof. Mag. DI DDr. Andreas Hauser
Institute of Experimental Physics

Co-Supervisor

DI Dr.techn. Thomas Hirsch
Institute of Software Technology

Graz, September 2024

Abstract

Physics-informed neural networks have pushed the boundaries of previously available methods by merging the strengths of purely analytical and purely data-driven methods. This hybrid technique boasts a wide array of applications, ranging from solving intricate mathematical problems to generating accurate weather forecasts. Within this work, we examined the potential of physics-informed neural networks in two topical research domains: the optimization of geothermal energy systems by the accurate prediction of borehole fluid outlet temperatures and the leveraging of potential energy predictions based on inexpensive models approximating the results of costly density functional theory calculations.

Heat pumps coupled to geothermal energy storage systems are highly energy efficient and have the potential to remarkably reduce the energy consumption of heating and cooling processes in buildings. However, to allow for optimal control of the heat pump, precise predictions of the temperature of the fluid leaving the borehole heat exchanger are required. Unfortunately, modeling the behavior of coupled boreholes in a borehole field is a highly complex task, demanding significant computational power. To address this issue, we aimed to create a powerful, yet computationally feasible physics-informed neural network that enhances the predictive capabilities of purely data-driven methods. For the implementation of our approach, we employ real-world data from a borehole field, connected to a heat pump system, in order to train a type of neural network specifically tailored for time series data. We extend our model by additional physics-based loss terms incorporated in the overall loss function to improve the model's predictive accuracy.
Three benchmark methods are tested, revealing that adding simple physical constraints to the loss function, indeed, helps to improve the prediction accuracy of our model. For the one-week forecast, the mean absolute error decreases from 0.60 °C to 0.53 °C through including the physical constraints, while for the two-week forecast, the mean absolute error decreases from 0.73 °C to 0.71 °C. Moreover, we compare our model to a complex, computationally intensive theoretical model, which outperforms our approach only slightly.

In the second part of this work, concerned with approximating Density Functional Theory calculations by physics-informed neural networks, we focus on Pt-Ni nanoclusters due to their significant role in catalysis and their status as well-studied systems by numerous research groups.
We start with inexpensive global geometry optimizations to explore the potential energy surface and calculate the corresponding energies with a cost-effective potential-based method. The most promising structures are then selected for Density Functional Theory calculations, which provide reference data for the training of our models.
For the physics-informed neural network in this approach, we employ a different ansatz

compared to our study on boreholes. Beginning with the purely theoretical, so-called embedded atom model, we incrementally replace terms in the corresponding equations with neural network expressions, thereby increasing the flexibility of the model. To ensure that fundamental physical laws are still respected, we add physical constraints to the loss function, attempting to guide the model's convergence in the right direction. Five different models are tested: a refitted version of the embedded atom model; a physics-informed network replacing the embedding function of the network with appropriate constraints in the loss function; a physics-informed neural network to replace the pair density function with its corresponding constraint; a model replacing both the embedding function and the pair density function by neural network expressions under incorporation of physics-based constraints in the loss function; and finally, a model replacing both embedding function and pair density function by neural network expressions but without the physics-based constraints. For the evaluation, we compare our five models against a purely theoretical embedding atom model with fixed parameter values taken from a standard table.
Our study reveals that replacing the embedding function in the embedded atom model can significantly increase the accuracy of the predictions. While the purely theoretical model yields a root mean squared error on our test set of 830.2 eV/atom for the energies, respectively 336.2 eV/Å for the forces, the physics-informed neural network achieves an error of 59.9 eV/atom for the energies and 212.1 eV/Å for the forces.

Kurzfassung

Physikbasierte neuronale Netzwerke haben die Grenzen bisheriger Methoden erweitert, indem sie die Stärken sowohl analytischer als auch datenbasierter Modelle vereinen. Diese hybriden Ansätze ermöglichen es, herausfordernde Probleme in unterschiedlichsten Bereichen zu lösen, von der Lösung komplexer mathematischer Gleichungen bis hin zu Wettervorhersagen. Im Zuge dieser Arbeit werden die Fähigkeiten physikbasierter neuronaler Netzwerke in zwei zukunftsweisenden Forschungsfeldern untersucht: der Optimierung von Erdwärmepumpen durch präzisere Vorhersagen der Temperatur des aus dem Erdwärmetauscher austretenden Fluids, und der Verbesserung der Genauigkeit kostengünstiger Modelle zur Vorhersage der Energien von Pt-Ni Nanoclustern, die ansonsten kostspielige Berechnungen mittels Dichtefunktionaltheorie erfordern.

Erdwärmepumpen sind effiziente und umweltfreundliche Systeme zur Heiz- und Kühltechnik, jedoch sind für ihre optimale Leistung möglichst genaue Vorhersagen über die Temperaturentwicklung in den Erdwärmetauschern erforderlich. Dies war bislang nur mit komplizierten, rechenaufwändigen Modellen möglich, da das Verhalten der Erdwärmetauscher von einer Vielzahl an Einflussfaktoren abhängt. Das Ziel dieser Arbeit ist es daher, mithilfe physikbasierter neuronaler Netzwerke ein hybrides Modell zu entwickeln, das präzise Vorhersagefähigkeiten bereitstellt, ohne dabei übermäßige Rechenkosten zu verursachen.
Für die Implementierung unserer Methode werden Daten einer realen Erdwärmepumpe verwendet, um unser Modell, das auf einem speziellen neuronalen Netzwerk für Zeitreihen-Daten basiert, zu trainieren. Physikalische Zusammenhänge werden durch mathematische Gleichungen dargestellt und in die Kostenfunktion des Modells integriert.
Zusätzlich werden drei verschiedene Referenzmodelle getestet, um die Leistung unserer Methode zu evaluieren. Es stellt sich heraus, dass wir durch die Integration einfacher physikalischer Bedingungen tatsächlich die Vorhersagefähigkeiten des Modells steigern und die durchschnittlichen absoluten Fehler von 0.60 °C auf 0.53 °C für 1-Wochen-Vorhersagen, und von 0.73 °C auf 0.70 °C für 2-Wochen-Vorhersagen reduzieren können. Darüber hinaus vergleichen wir unsere Ergebnisse mit einem ausgefeilten, rein theoretischen Modell und stellen fest, dass unsere Methode nicht wesentlich schlechter als dieses abschneidet, jedoch deutlich weniger Rechenressourcen erfordert.

Der zweite Teil der Arbeit widmet sich der Approximierung von Dichtefunktionaltheorie-Kalkulationen für Pt-Ni Nanocluster mithilfe von physikbasierten neuronalen Netzwerken. Pt-Ni Nanocluster spielen aufgrund ihrer hohen Reaktionsfähigkeit eine zentrale Rolle in Katalyse-Prozessen. Zudem wurden sie bereits von einer Vielzahl von Forschungsgruppen analysiert, wodurch wertvolle Referenzergebnisse zur Verfügung stehen. Der Ausgangspunkt dieser Arbeit sind globale Struktur-Optimierungen, wobei die zugehörigen Energien mithilfe einer kostengünstigen, potentialbasierten Methode berech-

net werden. Die vielversprechendsten Strukturen werden anschließend ausgewählt, um mittels Dichtefunktionaltheorie Referenzdaten zu berechnen, die die Basis für das Training unserer physikbasierten Modelle bilden. Für die Implementierung unseres physikbasierten Netzwerkes beginnen wir mit einem rein theoretischen, sogenannten eingebetteten Atommodell und ersetzen sukzessive Teile der Gleichung durch neuronale Netzwerke. Damit fundamentale physikalische Gesetzmäßigkeiten eingehalten werden, werden zusätzlich physikbasierte Bedingungen in die Kostenfunktion der Modelle eingebunden, die das Konvergenzverhalten in die richtige Richtung lenken sollen. Insgesamt werden fünf verschiedene Modelle getestet: Ein adaptiertes theoretisches Modell mit variablen Parametern, die an die Referenzdaten angepasst werden; ein physikbasiertes neuronales Netzwerk, bei dem die Embedding-Funktion durch ein neuronales Netzwerk mit entsprechenden zusätzlichen Bedingungen ersetzt wird; ein physikbasiertes neuronales Netzwerk, bei dem die paarweise Elektronendichte-Funktion durch ein neuronales Netzwerk mit zugehörigen Bedingungen in der Kostenfunktion ersetzt wird; ein Modell, bei dem sowohl die Embedding- als auch die paarweise Dichtefunktion durch neuronale Netzwerke mit entsprechenden physikalischen Bedingungen ersetzt werden; sowie abschließend ein Modell, bei dem sowohl die Embedding- als auch die paarweise Dichtefunktion durch neuronale Netzwerke ersetzt wird, allerdings ohne die Integration zusätzlicher Bedingungen in die Kostenfunktion.
Die Ergebnisse dieser Arbeit zeigen, dass das Modell, bei dem die Embedding Funktion durch neuronale Netze ersetzt wird, in der Lage ist, die Genauigkeit der Vorhersagen des rein theoretischen Modells deutlich zu übertreffen. Während das rein analytische Modell mit Parametern aus der Standardtabelle mittlere Fehler von 830.2 eV/Atom bzw. 336.2 eV/Å für die Energien bzw. Kräfte liefert, können mit dem physikbasierten Netzwerk die Fehler auf 59.9 eV/Atom für die Energien und 212.1 eV/Å für die Kräfte reduziert werden.

Contents

1. Introduction 13

2. Background 15
- 2.1. Physics-Informed Neural Networks (PINNs) . . . 15
  - 2.1.1. Neural Networks . . . 15
  - 2.1.2. Integrating Physics . . . 18
  - 2.1.3. General Formulation of PINNs . . . 19
  - 2.1.4. Time Series Forecasting . . . 20
  - 2.1.5. Evaluation Metrics . . . 22
- 2.2. Fundamentals of Thermal Energy Storages . . . 24
  - 2.2.1. Thermodynamics . . . 24
  - 2.2.2. Heat Pumps . . . 26
  - 2.2.3. Ground Source Heat Pumps . . . 27
- 2.3. Quantum Chemistry . . . 33
  - 2.3.1. Schrödinger Equation and Born-Oppenheimer Approximation . . . 33
  - 2.3.2. Geometry Optimization . . . 34
  - 2.3.3. DFT Basics . . . 35
  - 2.3.4. DFT Improvements / Exchange-Correlation Functionals . . . 38
  - 2.3.5. Basis Sets, Plane-Waves, Pseudopotentials . . . 40
  - 2.3.6. Computational Tools for Electronic Structure Calculations . . . 41

3. PINNs for Borehole Heat Exchangers 43
- 3.1. Experimental Setup . . . 43
  - 3.1.1. Dataset Description . . . 43
  - 3.1.2. Theoretical Model of the System . . . 45
  - 3.1.3. Dataset Analysis and Preprocessing . . . 45
- 3.2. Implementation . . . 50
  - 3.2.1. Implementation of the Baseline Methods: Constant and Linear Model 51
  - 3.2.2. Implementation of a Standard Autoregressive LSTM . . . 51
  - 3.2.3. Implementation of the Custom Loss . . . 51
  - 3.2.4. Hyperparameter Tuning . . . 54
  - 3.2.5. Evaluation Method . . . 55
- 3.3. Theoretical Reference Model . . . 56
  - 3.3.1. Long-term: g-function Model . . . 56
  - 3.3.2. Short-term: B2G Model . . . 57
- 3.4. Results . . . 60

3.5. Discussion 69
3.5.1. Limitations and Future Work 70

4. PINNs for Pt-Ni Nanoclusters **73**
4.1. Pt-Ni Nanoclusters 73
4.2. Neural Networks for PES Calculations 73
4.2.1. Embedded Atom Model (EAM) Potentials 74
4.2.2. SMATB Potentials 75
4.3. Implementation 75
4.3.1. Global Geometry Optimization 76
4.3.2. Sampling Procedure 76
4.3.3. Training Set Selection 77
4.3.4. DFT Training Data Generation 77
4.3.5. Refitted SMATB Model for Approximations of DFT Calculations . 79
4.3.6. PINNs for Approximating DFT Calculations 79
4.3.7. Evaluation Method 83
4.4. Results 84
4.5. Discussion 84

5. Conclusion **87**

A. Appendix A **89**

Bibliography **91**

List of Figures

2.1. Schematic visualization of a multilayer perceptron [16] 16
2.2. Schematic representation of a PINN [24] 20
2.3. Incorporation of temporal information in RNN models, taken from [27] . . 21
2.4. Iterative (a) and direct (b) multi-step forecasting methods, taken from [27] 22
2.5. Comparison of different cross-validation techniques 23
2.6. pV-Diagrams of the ideal (reversed) Carnot Cycle, taken from [35] 27
2.7. Ground-source heat pump schematic, taken from [36] 27
2.8. Schematic of borehole field with three borehole heat exchangers, taken from [39] 28
2.9. Schematic of common BHE designs, taken from [40] 29
2.10. Schematic of heat step functions and step pulses, taken from [40] 31
2.11. 5s-orbital of Ag with all-electron, small-core, or large-corre effective core potential. Image taken from [44]. 42

3.1. Schematic of the GSHP of the employed dataset from Ruiz-Calvo et al. [9] 44
3.2. Temperature sensor measurements in borehole 6, at various depths 46
3.3. Results of setting T_{out} equal to T_{in} for non-operating states longer than 3 hours 48
3.4. Inlet- and outlet temperatures of the whole time span 49
3.5. Plot of calculated $\dot{Q}_{\text{heat}}$ 54
3.6. Plot of calculated $k = \frac{1}{R_{\text{total}}} \cdot A$ 54
3.7. (a) 2D model and (b) 3D model of the thermal network model for the analyzed dataset[38] 58
3.8. Comparison of prediction results (mean R2 scores and standard deviations represented as error-bars) with and without missing value imputation for the standard ARLSTM (without custom loss) and the baseline models . . 60
3.9. Mean MAE scores over the 5 folds for experiment (1), with standard deviations represented as error bars 64
3.10. Mean MAE scores over the 5 folds for experiment (2), with standard deviations represented as error bars 64
3.11. Exemplary prediction plots for experiment (1) - Part 1 65
3.11. Exemplary prediction plots for experiment (1) - Part 2 66
3.12. Exemplary prediction plots for experiment (2) - Part 1 67
3.12. Exemplary prediction plots for experiment (2) - Part 2 68

4.1. Sampling results for 55-atom clusters 78
4.2. Logarithmic loss curve of the PINN for the embedding function over epochs 85

List of Tables

3.1. Original feature set . . . 46
3.2. Extended feature set for physics-based loss . . . 47
3.3. Encoded DateTime features . . . 49
3.4. Hyperparameter space for hyperparameter tuning. For parameter definitions refer to section 3.2 . . . 55
3.5. Split sizes for experiment 1, i.e. number of windows (each of total width = look-back window + prediction horizon = 2 weeks) for the subsets in the different folds . . . 56
3.6. Split sizes for experiment 2, i.e. number of windows (each of total width = look-back window + prediction horizon = 4 weeks) for the subsets in the different folds . . . 56
3.7. Standard settings for assessment of missing value Imputation . . . 60
3.8. R2 scores for missing value imputation . . . 61
3.9. Fixed parameters for hyperparameter tuning of experiment (1) . . . 61
3.10. Fixed parameters for hyperparameter tuning of experiment (2) . . . 62
3.11. Parameters used for the final evaluation of experiment (1) . . . 62
3.12. Parameters used for the final evaluation of experiment (2) . . . 63

4.1. SMATB parameters for Pt-Ni, taken from [70] . . . 76
4.2. Hyperparameters used for refitting the SMATB parameters . . . 80
4.3. Hyperparameter space used for grid search of PINN for embedding function 81
4.4. Hyperparameter space used for grid search of PINN for pair density function 82
4.5. Hyperparameter space used for grid search of PINN for both embedding and pair density function . . . 82
4.6. Hyperparameter space used for grid search of PINN for both embedding and pair density function without additional physical constraints . . . 83
4.7. Results for the validation (val.) set and test set for employing PINNs for approximated DFT calculations with the RMSE values for the energies E in meV/atom and the forces F in meV/Å. The results of the SMATB approach with the parameters from the standard table are taken from the study of R. Meyer [60]. . . . 84

1. Introduction

Modeling and predicting the behavior of complex systems using classical analytical relations often face severe challenges, introduced through issues such as missing boundary conditions, noise, and other sources of uncertainty. Over the past decades, machine learning has undergone tremendous progress, leveraging observational data to address problems that were previously deemed unsolvable. Despite this progress, machine learning methods often fail to generate physically consistent predictions, as extrapolation or observational biases result in inadequate generalization [1]. To overcome these limitations, the family of *physics-informed neural networks* (PINNs) has recently been introduced. PINNs allow for the integration of mathematical expressions of physical laws into machine learning frameworks, thereby effectively merging observational data with analytical relations to combine their strengths while mitigating their weaknesses.
Nowadays, PINNs have applications in a wide range of domains, spanning from classical problems such as solving the Navier-Stokes equations [2], over predicting weather patterns [3], to modeling financial market behavior [4]. Within this thesis, we will apply PINNs to two cutting-edge research topics: Potential Energy Surface calculations for Pt-Ni nanoclusters and temperature output predictions for thermal energy storage systems.

Thermal energy storage systems connected with heat pumps are highly promising facilities, in terms of decreasing the energy consumption of buildings. The main benefits of geothermal energy systems such as borehole heat exchangers pertain to their ability to provide a renewable source of energy. By using only a small amount of electric energy to extract or transfer energy to the ground, they provide efficient heating and cooling for buildings, while producing minimal greenhouse gas emissions. Given that buildings account for approximately 40% of global CO_2 emissions [5], they play a crucial role in combating global warming. The importance of reducing CO_2 emissions is emphasized by the fact that by May 2024, a complete year had elapsed during which each month has set a new record for the highest monthly global surface temperature (for the respective month) [6]. One of the main requirements for efficiently operating borehole heat exchangers is to have precise knowledge of the output temperature of the fluid. [7]. Improved accuracy in the output temperature predictions allows for better control of the system and greater energy savings. To determine the borehole outlet temperature, either analytical or numerical methods can be used. While numerical models are able to yield more accurate predictions, they also entail high computational efforts, thereby decreasing their utility in practical settings. On the other hand, purely analytical models typically rely on numerous assumptions, which reduces their prediction accuracy [8]. The aim of this work is to address these issues by using a combination of numerical (machine learning) and analytical (physical) methods, in order to create a model that yields satisfactory prediction performance while requiring only limited computational resources. To this end, we employ a type of PINN, specifically tailored for the realm of temperature output prediction for a borehole heat exchanger. In particular, we utilize a large dataset [9] of a real-world borehole system, in addition to domain knowledge and physical principles, to train a long-short-term memory (neural network type) on this temperature prediction task. Then, a detailed evaluation is performed, using diverse metrics and the results of a complex prediction model, developed by Ruiz-Calvo et al. [10] as reference. Through this methodology, we aim to find out whether PINNs improve the predictive accuracy and operational efficiency of borehole heat exchange systems.)

In the second part of this thesis we focus on molecular structure calculations for Pt-Ni nanoclusters. Alloy nanoclusters are particularly interesting heterogeneous metal-based catalysts, owing to their high reaction activity, robustness against pollution, and fine-tuning capabilities [11]. Since the performance of a catalyst strongly depends on its geometric structure, it is essential to investigate and accurately predict the most stable configurations and electronic properties of these nanoclusters before comparing their actual performance in catalytic reactions. While density functional theory (DFT) offers a well-suited approach for small aggregates, these calculations become infeasible for larger systems due to the exponential increase of the complexity with increasing number of electrons. Therefore, suggest a novel approach, combining purely physics-based models with neural network expressions, in order to achieve a powerful PINN that accurately captures the complex patterns of the atomic interactions while significantly reducing the computational requirements. In particular, we gradually replace parts of the purely physics-based model by small neural networks obeying additional physical constraints. The models are fitted to costly DFT predictions and evaluated by means of the root mean squared error (RMSE) for the model predictions on the testing data. Furthermore, we benchmark our models against the original, physical model to provide a thorough assessment of their predictive capabilities.

This thesis is structured as follows: After a brief recap of the theoretical foundations required to understand the methods employed in this work, we will progress to the exploration of the topic of thermal energy storage systems with its practical applications of borehole heat exchangers and heat pumps. Subsequently, we will delve into the quantum chemistry part which is concerned with approximating DFT calculations for potential energy surfaces of Pt-Ni nanoclusters, before we ultimately reach the final conclusion of our studies on PINNs for different applications.

2. Background

This chapter is dedicated to the theoretical foundations of this work. It is vital to obtain an in-depth understanding of the theory and concepts, in order to grasp the applied methods in a practical experiment setting. The chapter starts with a basic introduction to physics-informed machine learning, combining the latest advancements in computer science with fundamental physical laws. Then, the thermodynamic equations required to roughly model thermal energy storage will be explained. These insights will be necessary to integrate physical relations into the neural network-based predictions for such systems. Thereafter, we will delve into quantum chemistry, giving a brief introduction to Density Functional Theory (DFT), in particular Kohn-Sham DFT with generalized gradient approximation methods, followed by a brief introduction to computational tools for electronic structure calculations.

2.1. Physics-Informed Neural Networks (PINNs)

Deep learning techniques have developed rapidly over the last decade. However, training these algorithms typically requires significant amounts of data and computation time. Physics-informed machine learning aims to embed additional physical information to neural networks or other kernel-based regression networks to achieve enhanced accuracy even with limited training data, reduced training (computation) time, and better generalization of the algorithm [1]. The subsequent section starts with a general introduction to deep learning, specifically focusing on neural networks, followed by an exploration of the integration of physics such as partial differential equations (PDEs) into deep neural network expressions. Finally, we will introduce physics-informed neural networks (PINNs) specifically tailored for computational chemistry.

2.1.1. Neural Networks

The term "deep learning" (DL) refers to a subset of machine learning techniques where algorithms accumulate knowledge due to analysis of large amounts of data [12]. DL algorithms comprise numerous components, such as linear algebra, probability theory, calculus, optimization, programming, signal processing, and high-performance computing. As the techniques applied originally derive from neural networks (NNs), the attribute "deep" characterizes models of long compositions, i.e. number of layers [13]. (Artificial) neural networks are inspired by the biological architecture of the brain. They were first introduced in 1943 [14] in the form of a simplified computational model of neurons in animal brains. An advancement of this first concept was introduced by Frank Rosenblatt with the *Perceptron*, and then extended to the Multilayer Perceptron [15].

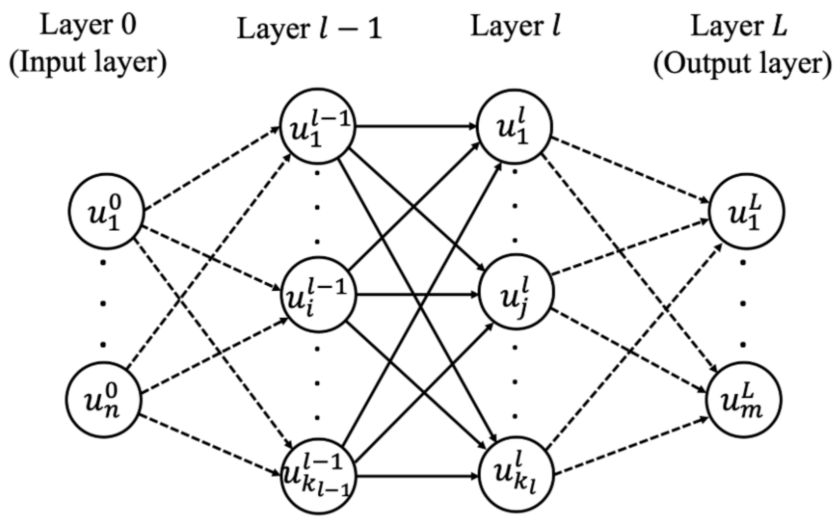


Figure 2.1.: Schematic visualization of a multilayer perceptron [16]

Multilayer Perceptron

A multilayer perceptron consists of a total number of $L+1$ layers indexed with l, each containing a number of artificial neurons, denoted as k_l, or as n in the input layer $l=0$. These neurons, also referred to as units u_k^l, account for mathematical operations given as weighted sum of their inputs in addition to a bias term b, which is then transformed by a non-linear activation function σ. Each neuron in a layer l_i is connected to each neuron in the prior layer l_{i-1}, with i ranging from $1 \dots L$. The connections between the individual neurons are weighted by factors w_{l_i}. Thus, the output of the k-th neuron in layer l is given as [16]:

$$u_k^l = \sigma\Big(\sum_{j=1}^{k_l-1} w_{k,j}^l u_j^{l-1} + b_k^l\Big). \tag{2.1}$$

The layers from $i = 1...L-1$ are called *hidden layers.* We can rewrite the equation in compact matrix form by dropping the subscripts:

$$\mathbf{u}^l = \sigma(\mathbf{W}^l \mathbf{u}^{l-1} + \mathbf{b}^l) \tag{2.2}$$

Among the most commonly used activation functions σ are the logistic (sigmoid) function, the hyperbolic tangent, and the rectified linear unit [16]. A visual representation of a multilayer perceptron is given in Figure 2.1.

Multilayer perceptrons (MLPs) represent the most traditional architecture of deep learning methods. It is also known as fully-connected neural network [17]. An MLP with more than one hidden layer is called deep neural network (DNN). The main advantage of neural networks is their ability to represent any bounded continuous function, given only a single hidden layer and a finite number of neurons (universal approximation theorem [18]). Hence, the goal of applying neural networks to some input data is to find the parameters (i.e. weights, biases) required for the representation of the input. To this end, *automatic differentiation* (AD) or its generalization for deep learning, called "backpropagation", is commonly employed to obtain the derivatives of the outputs of the network with respect to its inputs [16]. The backpropagation training algorithm starts with feeding the input (training) instances to the neurons of the first hidden layers, which compute the output according to Eq. 2.1 and pass it to the neurons of the next layer. This is repeated for every neuron in each consecutive layer. Consequently, this process is called *forward pass.* The network output can then be compared to the true output (provided by the labeled training instances), in order to compute to which extent each neuron of the last hidden layer contributed to the mismatch between the true and the predicted output (loss). The algorithm then proceeds to the previous layer and computes how much each of those neurons contributed to the error. This process is known as *reverse pass.* Finally, a *gradient descent* step is performed, which adapts the weights and biases of the network to decrease the loss. This whole process is also referred to as *backpropagation* algorithm. MLPs can be used for both regression and classification. In the case of a classification task, each output corresponds to one class. By applying a softmax function to the neurons of the output layer, each output neuron yields an estimated probability for its corresponding class [15]. The softmax function transforms the raw output scores (logits) into probabilities by applying:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_i}}. \tag{2.3}$$

Recurrent Neural Networks

Recurrent neural networks (RNNs) are a special type of neural network with a modified architecture compared to the standard MLP, resulting in a different way of passing information through the network. In contrast to an MLP, the RNN incorporates cycles, allowing for a transmission of information back into itself. Thus, not only the current input X_t but also previous inputs $X_0, X_1, \ldots, X_{t-1}$ can be considered. The architecture of RNNs makes them particularly suitable for the detection of patterns in sequential data, such as text, or numerical time series. To correctly backpropagate the loss (reverse pass) in time, RNNs employ a slightly adapted backpropagation algorithm, known as backpropagation through time (BPTT). However, RNNs frequently face the problem of vanishing gradients, which arise in long sequences with small values of in the matrix multiplications through the layers of the network. The vanishing gradient problem occurs during backpropagation when the gradients of the loss function with respect to the network's weights become progressively smaller as they are propagated backward through the layers of the network. This happens because, in RNNs, the same weights are used repeatedly across time steps, and when these weights are involved in a series of multiplications, especially if they are less than one, the gradients shrink exponentially. This diminishes the influence of states that happened far before the current time step, thereby making it difficult for the network to learn long-term dependencies. Conversely, very large values in the matrix multiplications can lead to exploding gradients, producing excessively large weights for earlier time steps [19].

Long-short-term memory

In 1997, Hochreiter & Schmidhuber [20] introduced Long-short-term memory (LSTM) units, aiming to overcome the vanishing gradient problem by storing additional information outside the traditional neural network in special gated cells [19]. These cells use multiplicative gate units that control the information flow by learning when to open and close in order to determine what data enters and exits the memory cell(s). The memory cells comprise a recurrently self-connected linear unit, which is also referred to as "constant error carousel". These carousels address the vanishing gradient problem by maintaining consistent local error propagation unless new signals (inputs) arrive. This allows the LSTM to selectively discard or retain information, according to what is required [21].

A more detailed explanation of LSTMs, specifically applied for time series forecasting, is provided in subsection 2.1.4.

2.1.2. Integrating Physics

Neural networks that solely rely on data often run into the problem of achieving reasonable performance at fitting observations, but fail at generalizations due to extrapolation biases. In addition, they typically struggle in tasks with few or noisy training data. Physics-informed learning is a promising approach, aimed at leveraging the performance of ML algorithms by integrating domain knowledge and fundamental physical laws into data-driven models [2, 1]. To embed physics into a learning algorithm, there are currently three pathways, directing the output toward physically consistent solutions:

- *Observational biases*: Introducing an observational bias refers to using data that already embody the underlying physics so that algorithms trained on this data automatically capture the functions and operators reflecting the physical structure in the data. This type of method represents the simplest way of introducing a physics-informed bias [1].

- *Inductive biases*: Inductive biases, on the contrary, pertain to specific NN architectures, tailored in a way that predictions are guaranteed to satisfy certain physical constraints, typically given as mathematical equations. The most prominent NN architecture with an inductive bias is represented by convolutional neural networks (CNNs) [22]. These networks are predominantly used in computer vision, as they preserve invariances across groups of symmetries and distributed pattern representations, which are inherent to natural images. Further examples of inductive biases include graph neural networks, kernel methods (e.g. Gaussian processes), and equivariant networks [1].

- *Learning biases*: In place of enforcing prior knowledge through a specific architecture, learning biases are employed through penalty terms in the loss function of conventional NN architectures. This way, the physical constraints are enforced as soft constraints. The learning algorithm ought to simultaneously fit the training data and fulfill a given set of physical equations, e.g. conservation of mass and momentum. While due to the soft manner of the implemented constraints, the underlying laws are, in general, only approximately satisfied, these soft constraints at the same time enable substantial versatility, so that diverse biases can be introduced. Strictly speaking, *physics-informed neural networks* (PINNs), which were first introduced by Raissi et al.[23] in 2019, are traditional neural networks, with modified loss functions, thus the physical foundations are solely reflected in the training process of the method, but not in the architecture.

These three ways of incorporating physical laws into data-driven ML algorithms can also be combined, yielding a large variety of hybrid approaches [1].

2.1.3. General Formulation of PINNs

To obtain a general formulation of PINNs in the strict sense (i.e. based on learning biases), we assume that the underlying physical laws we aim to include in our MLP as additional knowledge, are given in form of a parameterized partial differential equation (PDE):

$$\begin{aligned} f(\mathbf{x}, t, \hat{u}, \partial_x \hat{u}, \partial_t \hat{u}, \ldots, \lambda) = 0,\ \mathbf{x} \in \Omega, t \in [0, T] \\ \hat{u}(\mathbf{x}, t_0) = g_0(\mathbf{x}),\ \mathbf{x} \in \Omega, \\ \hat{u}(\mathbf{x}, t) = g_\Gamma(t),\ \mathbf{x} \in \partial\Omega, t \in [0, T]. \end{aligned} \tag{2.4}$$

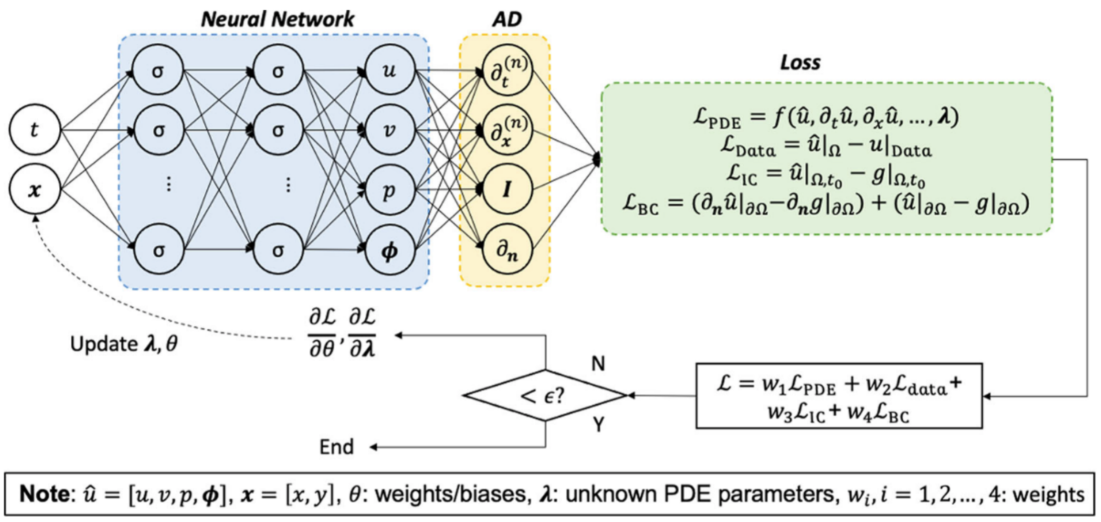
t denotes the time, $x \in \mathbb{R}^d$ the spatial coordinate, and f accounts for the residual of the PDE, containing the differential operators and the parameters λ. The solution of the PDE is given as $\hat{u}(\mathbf{x}, t)$ with the initial/boundary conditions denoted as $g_0(\mathbf{x})$, and $g_\Gamma(t)$, respectively. Ω is the spacial domain and $\partial\Omega$ the boundary. The boundary conditions can be of either type: Dirichlet, Neumann, or mixed [24].

Given space and time coordinates $\mathbf{x}, t$ as inputs, the PINN aims to approximate the solution of the PDE by simultaneously learning both the parameters of the network (weights, biases) and the parameters of the PDE. To this end, the objective (loss function) is transferred into an optimization problem of the following form:

$$L = \omega_1 L_{PDE} + \omega_2 L_{data} + \omega_3 L_{IC} + \omega_4 L_{BC}, \tag{2.5}$$

where ω_i are coefficients allowing for a weighting of the individual loss terms, L_{PDE} represents a penalty for the residuals of the PDE, L_{data} gives the loss due to the mismatch between model predictions (outputs) and the true values, and $L_I C$ respectively $L_B C$ account for the initial and boundary conditions. Typically, the mean square error is used to compute the individual loss terms. To minimize the loss function by iteratively updating the parameters, ADAM [25] optimizer is most commonly used. A schematic of a general PINN is shown in Figure 2.2.

In this illustration, the a neural network with inputs given by space $\mathbf{x}$ and time t coordinates is employed to solve for $\hat{u}$. The model uses automatic differentiation (AD) to calculate the derivatives of $\hat{u}$ with respect to the inputs. These derivatives are further used to calculate the residuals of the physics-based loss terms. The PINN is thereby able to simultaneously learn the weights and biases of the neural network, as well as the unknown parameters of the PDE [24].



Neural Network
AD
Loss
t
x
σ
u
v
p
ϕ
$\partial_t^{(n)}$
$\partial_x^{(n)}$
I
∂_n
$\mathcal{L}_{\mathrm{PDE}} = f(\hat{u}, \partial_t \hat{u}, \partial_x \hat{u}, \ldots, \boldsymbol{\lambda})$
$\mathcal{L}_{\mathrm{Data}} = \hat{u}|_{\Omega} - u|_{\mathrm{Data}}$
$\mathcal{L}_{\mathrm{IC}} = \hat{u}|_{\Omega,t_0} - g|_{\Omega,t_0}$
$\mathcal{L}_{\mathrm{BC}} = (\partial_n \hat{u}|_{\partial\Omega} - \partial_n g|_{\partial\Omega}) + (\hat{u}|_{\partial\Omega} - g|_{\partial\Omega})$
Update $\boldsymbol{\lambda}, \theta$
$\frac{\partial \mathcal{L}}{\partial \theta}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}}$
N
$< \epsilon?$
Y
End
$\mathcal{L} = w_1 \mathcal{L}_{\mathrm{PDE}} + w_2 \mathcal{L}_{\mathrm{data}} + w_3 \mathcal{L}_{\mathrm{IC}} + w_4 \mathcal{L}_{\mathrm{BC}}$
Note: $\hat{u} = [u, v, p, \boldsymbol{\phi}]$, $\boldsymbol{x} = [x, y]$, θ: weights/biases, $\boldsymbol{\lambda}$: unknown PDE parameters, $w_i, i = 1, 2, \ldots, 4$: weights

Figure 2.2.: Schematic representation of a PINN [24]

2.1.4. Time Series Forecasting

The term *time series* pertains to a time-ordered sequence of values. I.e., a univariate time series X, where the value at each time step t_i is given by a single real number, can be expressed as:

$$X = [x_1, x_2, \ldots, x_T], \tag{2.6}$$

with T denoting the total number of time steps. Naturally, a multivariate time series is a time-ordered sequence of vectors, each containing M real numbers.

$$\mathbf{X} = [X^1, X^2, \ldots, X^M], \tag{2.7}$$

which can also be considered as M univariate time series with $X^i \in \mathbb{R}^T$ [26]. In time series forecasting, the aim is to predict the future values of a target y_t at time t. One-step forecast models can then be expressed as:

$$\hat{y}_{t+1} = f(y_{t-k:t}, \mathbf{x}_{t-k:t}, \mathbf{s}), \tag{2.8}$$

where $f(.)$ is the learned prediction function, $\hat{y}_{t+1}$ represents the forecast output of the model, $y_{t-k:t} = \{y_{t-k}, \ldots, y_t\}$ are past observations of the target, and $\mathbf{x}_{t-k:t} = \{\mathbf{x}_{t-k}, \ldots, \mathbf{x}_t\}$ represent the past external inputs over a look-back window k. In addition, s accounts for static metadata [27]. In neural networks for time series tasks, predictive relationships are learned by extracting relevant historical information through a series of nonlinear layers. These layers encode the information into a latent variable $\mathbf{z}_t$. Thus, the forecast function can be expressed as:

$$f(y_{t-k:t}, \mathbf{x}_{t-k:t}, \mathbf{s}) = g_{\mathrm{dec}}(z_t), \tag{2.9}$$

with

$$z_t = g_{\mathrm{enc}}(y_{t-k:t}, \mathbf{x}_{t-k:t}, \mathbf{s}), \tag{2.10}$$

where $g_{\text{enc}}(.)$ and $g_{\text{dec}}(.)$ represent encoder, respectively decoder functions, whose exact type depends on the chosen network architecture [27]. For RNN models such as the LSTM, the temporal information is encoded by a recurrent layer that contains the temporal information of the past (look-back window). For each future output prediction at time step t, the model is provided with the external input data x_t and the information of the past. A visualization is given in Figure 2.3.

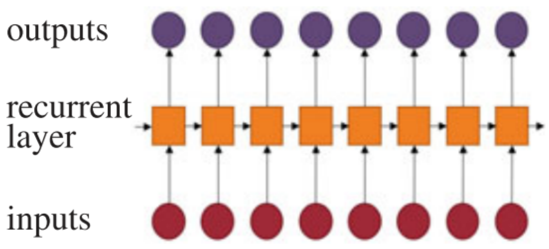


Figure 2.3.: Incorporation of temporal information in RNN models, taken from [27]

RNN cells are characterized by an internal memory state that captures previous (historical) information. At each new time step, this memory state $z_t$is recursively updated in the way:

$$\mathbf{z}_t = \nu(\mathbf{z}_{t-1}, y_t, \mathbf{x}_t, \mathbf{s}), \tag{2.11}$$

with $\nu(.)$ being the learned memory update function. Therefore, a simple RNN variant for time series forecasting can be expressed as:

$$y_{t+1} = \gamma_y(\mathbf{W}_y \mathbf{z}_t + \mathbf{b}_y), \tag{2.12}$$

with

$$\mathbf{z}_t = \gamma_z(\mathbf{W}_{z_1} z_{t-1} + \mathbf{W}_{z_2} y_t + \mathbf{W}_{z_3} \mathbf{x}_t + \mathbf{W}_{z_4} \mathbf{s} + \mathbf{b}_z). \tag{2.13}$$

Here, $\mathbf{W}$ and $\mathbf{b}$ denote the weights and biases, and $\gamma_y(.), \gamma_z(.)$ represent the activation functions. As mentioned in subsubsection 2.1.1, basic variants of RNNs typically struggle at learning long-range dependencies because of the infinite look-back window that leads to exploding or vanishing gradients. Therefore, the more sophisticated RNN variant of LSTM was developed to overcome this issue by using cell states c_t that store long-term information, regulated by a set of specialized gates, including the input gate, forget gate, and output gate. The goal of the input gate is to control the amount of new information that enters the cell state, while the forget gate determines which portion of past information is discarded, and the output gate governs the influence of the cell state on the current output. Mathematically, these gates can be expressed as:

$$\begin{aligned} \text{input gate: } \mathbf{i}_t &= \sigma(\mathbf{W}_{i_1} \mathbf{z}_{t-1} + \mathbf{W}_{i_2} y_t + \mathbf{W}_{i_3} \mathbf{x}_t + \mathbf{W}_{i_4} \mathbf{s} + \mathbf{b}_i) \\ \text{output gate: } \mathbf{o}_t &= \sigma(\mathbf{W}_{o_1} \mathbf{z}_{t-1} + \mathbf{W}_{o_2} y_t + \mathbf{W}_{o_3} \mathbf{x}_t + \mathbf{W}_{o_4} \mathbf{s} + \mathbf{b}_o \\ \text{forget gate: } \mathbf{f}_t &= \sigma(\mathbf{W}_{f_1} \mathbf{z}_{t-1} + \mathbf{W}_{f_2} y_t + \mathbf{W}_{f_3} \mathbf{x}_t + \mathbf{W}_{f_4} \mathbf{s} + \mathbf{b}_f, \end{aligned} \tag{2.14}$$

where $\sigma(.)$ is the sigmoid activation function and $\mathbf{z}_{t-1}$ represents the hidden state of the LSTM, given as:

$$\mathbf{z}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{2.15}$$

with $\odot$ being the element-wise product. The cell state c_t can be expressed as [27]:

$$\mathbf{c}_t = \mathbf{f}_t \odot c_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{c_1}\mathbf{z}_{t-1} + \mathbf{W}_{c_2}\mathbf{x}_t + \mathbf{W}_{c_4}\mathbf{s} + \mathbf{b}_c). \tag{2.16}$$

Forecasting future targets that are continuous corresponds to a regression task. Thus, for one-step forecast models, the most common loss function is given by the mean square error loss

$$\mathcal{L}_{\text{MSE}} = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2. \tag{2.17}$$

The one-step forecasting models discussed so far, can straightforwardly be extended to multi-step forecasts of the form:

$$\hat{y}_{t+\tau} = f(y_{t-k:t}, \mathbf{x}_{t-k:t}, \mathbf{u}_{t-k:t+\tau}, \mathbf{s}, \tau), \tag{2.18}$$

where $\tau \in \{1, \ldots, \tau_{\max}\}$ denotes the discrete prediction (forecast) horizon, $\mathbf{x}_t$ are external inputs that are only available for historical time steps, and $\mathbf{u}_t$ are external inputs that are given for future time steps, such as date- and time- information. Multi-step forecasting methods can be divided into recursive (also called iterative) approaches and direct approaches. Recursive methods usually employ autoregressive architectures, which generate forecasts by recursively feeding target samples into subsequent future time steps. In contrast, direct methods use all provided inputs (including past and a priori inputs) simultaneously and produce a fixed-length output vector according to the desired prediction horizon. A visualization of these two different methods is provided in Figure 2.4 [27].

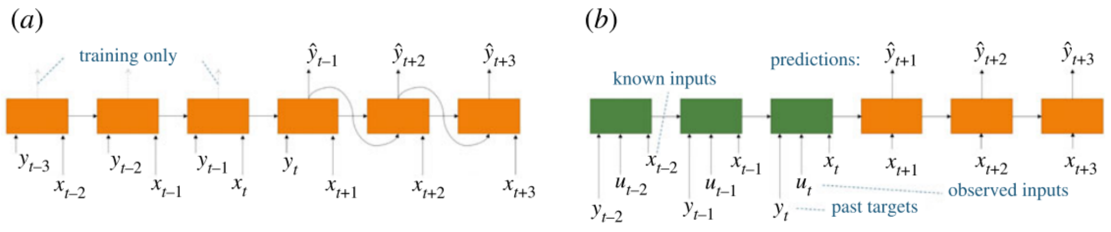


Figure 2.4.: Iterative (a) and direct (b) multi-step forecasting methods, taken from [27]

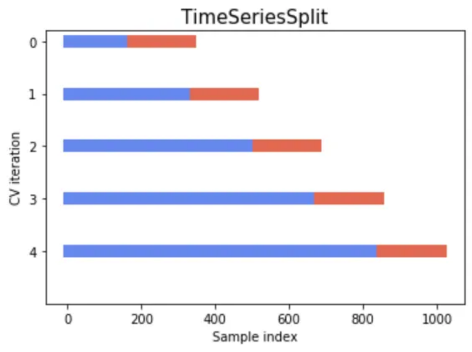
2.1.5. Evaluation Metrics

To evaluate the results of a time series forecast with continuous targets, which equals a regression task, several performance metrics can be considered. Some of the most widely used metrics for regression are:

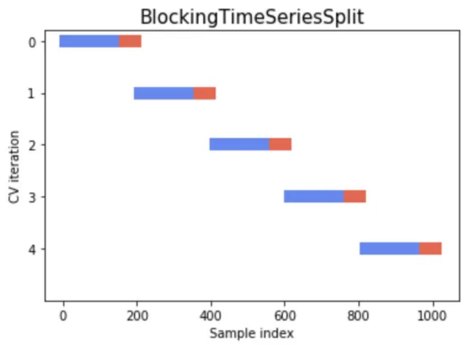
$$\begin{aligned}\text{Mean Absolute Error (MAE): MAE} &= \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t| \\ \text{Root Mean Squared Error (RMSE): RMSE} &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \\ \text{Coefficient of determination (R2 score): R2} &= 1 - \frac{SS_{res}}{SS_{tot}},\end{aligned} \tag{2.19}$$

where $SS_{res} = \sum_{t=1}^{T}(y_t - \hat{y}_t)^2$ is the sum of squared residuals and $SS_{tot} = \sum_{t=1}^{T} = (y_t - \bar{y})^2$ is the total sum of squares with $\bar{y}$ being the mean of the target data y_t.

Due to the inherent correlation of time series data, it is vital to avoid information leakages during the evaluation procedure. Thus, different methods of time series cross-validation have been developed, such as blocked cross-validation and time series split cross-validation [28]. Time series split cross-validation is based on the property that the validation set is always ahead of the training set so that the model does not predict the past. Therefore, in each iteration, the corresponding time series split (fold) is divided into a training, and a validation set. If hyperparameter tuning is performed (on the validation set), the individual splits can also be divided into the three subsets training, validation and testing set in each iteration. Furthermore, in each subsequent iteration, the previous validation (and testing) set, become part of the training set, and the following sequence is selected as validation set. Thus, the training set size increase in each subsequent iteration (fold). A visualization of this method is given in Figure 2.5a. Another approach is blocked time series cross-validation, where it is ensured that each fold has the same size. To this end, the time series splits are shifted across the entire sequence as sliding windows, where again, for each iteration the split is divided into training and validation (and potentially testing) set [28]. A schematic is given in Figure 2.5b.



(a) Schematic of time series split cross-validation, taken from [28]



(b) Schematic of blocked time series cross-validation, taken from [28]

Figure 2.5.: Comparison of different cross-validation techniques

2.2. Fundamentals of Thermal Energy Storages

Thermal energy storage refers to systems that store energy through cooling or heating a certain storage medium. The aim is to enable using the stored thermal energy for heating or cooling purposes at a later time when there is a demand for it [29]. It was estimated that through more extensive usage of heat and cold storage systems, roughly 1.4 million GWh energy and 400 Mt CO2 emissions per year could be saved in Europe [30]. Thermal energy storage ranges from water tank storage over chemical phase-change storage up to underground storage [29]. Underground storage systems offer the advantage that they accommodate substantial volume under confined surface areas and are therefore a widely used technique. The key concept of underground storage is to make use of the ground, comprising soil, rocks, clay, etc., as a storage medium. Among the most prominent types of underground thermal energy storage systems are source heat pumps, which will be explained in the following section.

2.2.1. Thermodynamics

The most important principles of Thermodynamics can be summarized by three fundamental laws: The so-called zeroth law, which defines a system to be in a thermal equilibrium if the temperature is the same throughout the system and it is equal to the temperature of its surroundings. The first law of thermodynamics, which is essentially just a formulation of the law of conservation of energy, can be written as:

$$\Delta U = Q - W, \tag{2.20}$$

where Q is the heat supplied or extracted from the system, W is the work done on or by the system, and ΔU represents the change in internal energy. The second law of thermodynamics defines the direction of heat flow, in particular, it states that heat can spontaneously only flow from a reservoir of higher temperature to lower temperature [31]. With these fundamental laws being given, one can already calculate the heat transferred from one equilibrium state to another. Nevertheless, to calculate the rate of heat transfer or variation of the temperature with time and space, additional equations are needed: the laws of heat transfer. These laws extend the basics of thermodynamics by fluid flow and rate equations.
Heat transfer can be divided into the three different modes conduction, convection, and radiation. Conduction is a process where the energy of the particles in a substance (can be either liquid, solid, or gaseous) is transferred from the higher energetic particles to their less energetic neighbors. For solids, this happens in the form of lattice vibrations and/or free electrons, while in gases and liquids, the energy is transferred through the collisions of molecules. The main equation for the conduction process is Fourier's law, which states that "The rate of heat flow by conduction in a given direction is proportional to the area normal to the direction of heat flow and to the gradient of temperature in that direction." [32] For the example of heat flux in x-direction $\dot{Q}_x$, this translates to

$$\dot{Q}_x = -kA\frac{dT}{dx}, \tag{2.21}$$

where k is a proportionality constant, also called thermal conductivity, A is the cross-sectional surface area of the material the heat is transferred through, and $\frac{dT}{dx}$ is the temperature differential in transfer direction. More generally, the heat conduction equation is given as:

$$\frac{\partial T(\mathbf{x},t)}{\partial t} = a\Delta T(\mathbf{x},t), \tag{2.22}$$

where a is the conductivity of the material [33].
Convection, on the other hand, describes the heat transfer that occurs when a fluid flows over a surface that has a different temperature than itself. In addition, convection also takes place when two fluids with different temperatures are mixed. The governing equation the heat flux $\dot{Q}$ for convection is given by Newton's law of cooling:

$$\dot{Q} = hA(T_s - T_f), \tag{2.23}$$

where h is the heat transfer coefficient for convection, T_f is the fluid temperature, T_s is the surface temperature and A represents the surface area that is exposed to the fluid.

Thermal radiation describes the radiation that is emitted at a wavelength between λ = 0.1µm - 100µm. The main equation for radiation energy flux E_b emitted from an ideal emitter (black body) is given by Stefan-Boltzmann's law:

$$E_b = \sigma T^4, \tag{2.24}$$

where $\sigma = 5.6697 \cdot 10^{-8}$ W/m^{2}K is known as the Stefan-Boltzmann constant and T represents the temperature of the radiating body [32].

In order to analyze thermal processes, it is often helpful to establish an energy balance that accounts for all parts of energy that enter or exit the investigated system. The general energy balance, as introduced by the first law of thermodynamics Equation 2.20, can be reformulated for rates in steady-state open systems by:

$$\sum_{\text{out}}(\dot{m}\hat{E})_{\text{out}} - \sum_{\text{in}}(\dot{m}\hat{E})_{\text{in}} = \dot{Q} - \dot{W}, \tag{2.25}$$

where $\dot{m}$ gives the mass flow rate of a stream, $\hat{E}$ corresponds to the energy per mass, $\dot{Q}$ is the rate of heat transferred into the system or generated within the system and $\dot{W}$ is the work done on a system per time. The process of heating or cooling a material without exhibiting a phase change is called sensible heating (or cooling). For systems with negligible change in potential energies and without external work, the energy balance can then be reformulated as:

$$\sum_{\text{out}}\Big(\dot{m}\bar{c_p}(T - T_{\text{ref}})\Big)_{\text{out}} - \sum_{\text{in}}\Big(\dot{m}\bar{c_p}(T - T_{\text{ref}})\Big)_{\text{in}} = \dot{Q}, \tag{2.26}$$

with $\bar{c_p}$ is the specific heat capacity of the material averaged between the reference temperature T_{ref} and the corresponding temperature of the output/input stream T [34].

For a single inlet and outlet stream, such as given in a pipe system of a borehole heat exchanger, the reference temperatures cancel out and the equation simplifies to

$$\dot{Q} = c_p \dot{m} |T_{\text{out}} - T_{\text{in}}|, \tag{2.27}$$

where c_p is the specific heat capacity of the material at constant pressure for a given temperature of the system, T_{out} is the outlet temperature, T_{in} the inlet temperature, and $\dot{Q}$ the heat extracted from the system.

2.2.2. Heat Pumps

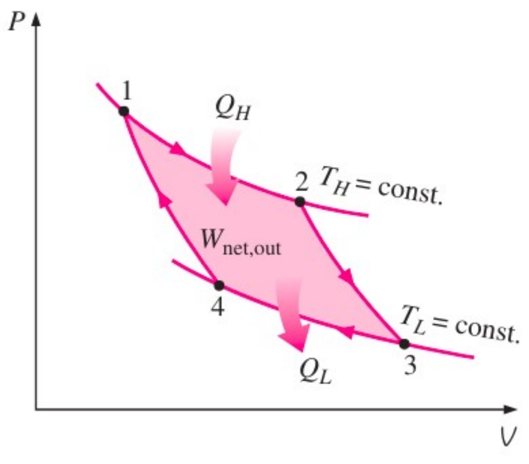
Heat pumps are devices that transfer heat from reservoirs of low temperature to reservoirs of higher temperature. In accordance with the second law of thermodynamics, this process cannot happen spontaneously but requires input of external work. The main goal of a heat pump is to maintain a heated space at a certain temperature, which can be achieved by absorbing heat from a material such as water or air. This divides heat pumps into two different types: water-to-water and water-to-air heat pumps. The performance measure of heat pumps is called coefficient of performance, which can be expressed as

$$COP_{\text{HP}} = \frac{Q_H}{W_{\text{in}}} = \frac{1}{1 - Q_L/Q_H}, \tag{2.28}$$

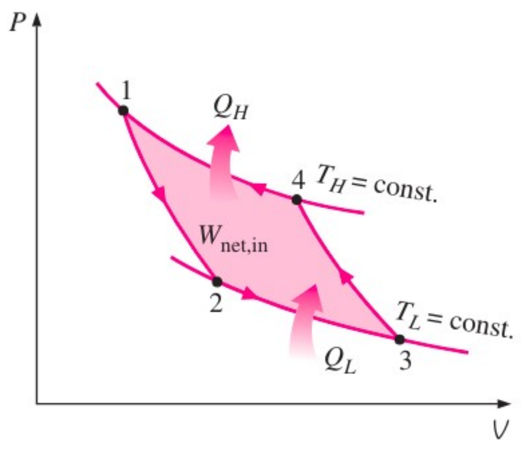
where W_{in} gives the net input of work Q_H is the magnitude of heat transferred to the warm environment at high temperature T_H, and Q_L is the amount of heat extracted from the region at low temperature T_L.
A fundamental principle of thermodynamics is the Carnot Principle, which states that there is no heat engine operating between two heat reservoirs that is more efficient than a reversible Carnot heat engine. A Carnot heat engine is a heat engine operating on the Carnot cycle, which is composed of two isothermal (constant temperature) and two adiabatic (no heat exchange) processes. A diagram visualizing the Carnot cycle is given in Figure 2.6a. A Carnot heat pump is the reverse of a Carnot heat engine (compare Figure 2.6b). In the first sub-process of the Carnot heat pump cycle, heat Q_L is absorbed from a reservoir of low temperature T_L, then the working medium is adiabatically compressed such that its temperature increases to T_H. In the next step, the heat Q_H is released to the warmer environment, and finally, the medium is adiabatically expanded until its temperature reaches the initial low temperature T_L again. For the ideal Carnot heat pump, the COP can be expressed as:

$$COP_{\text{HP, Carnot}} = \frac{1}{1 - T_L/T_H}, \tag{2.29}$$

which defines an upper limit for any real heat pump system, operating between reservoirs of temperature T_L and T_H [35].



(a) Carnot Cycle

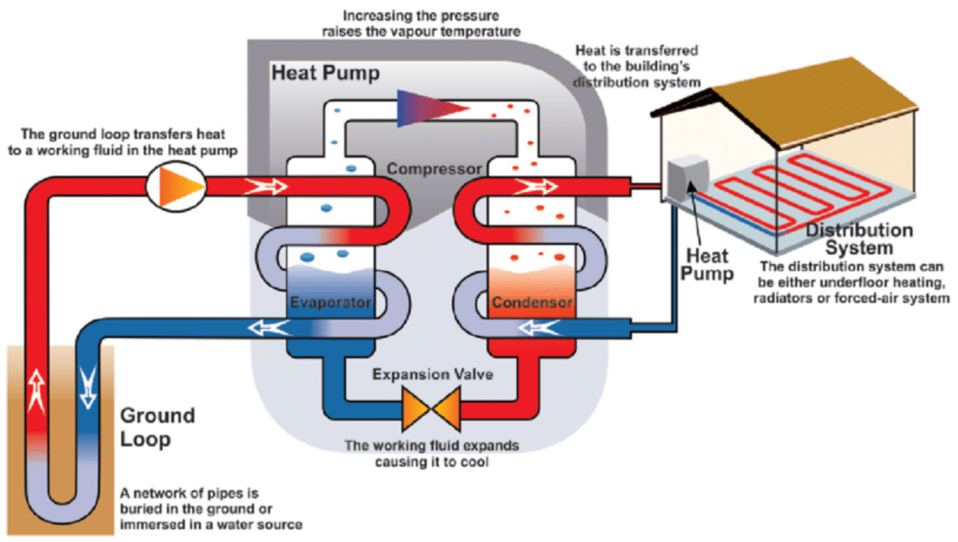


(b) Reversed Carnot Cycle

Figure 2.6.: pV-Diagrams of the ideal (reversed) Carnot Cycle, taken from [35]

2.2.3. Ground Source Heat Pumps

Ground source heat pumps (GSHP) are highly energy-efficient heating and cooling systems. In winter, they make use of the heat stored in the ground, while in summer they harness the cooler temperatures of the ground as a sink for excess heat. This heat transfer process is accomplished by borehole heat exchangers (BHEs), which are the underground system that allows the GSHP to exchange heat with the ground.



Increasing the pressure raises the vapour temperature
Heat Pump
Heat is transferred to the building's distribution system
The ground loop transfers heat to a working fluid in the heat pump
Compressor
Evaporator
Condensor
Distribution System
The distribution system can be either underfloor heating, radiators or forced-air system
Heat Pump
Expansion Valve
Ground Loop
The working fluid expands causing it to cool
A network of pipes is buried in the ground or immersed in a water source

Figure 2.7.: Ground-source heat pump schematic, taken from [36]

The most common type of borehole heat exchanger systems is the *vertical borehole*

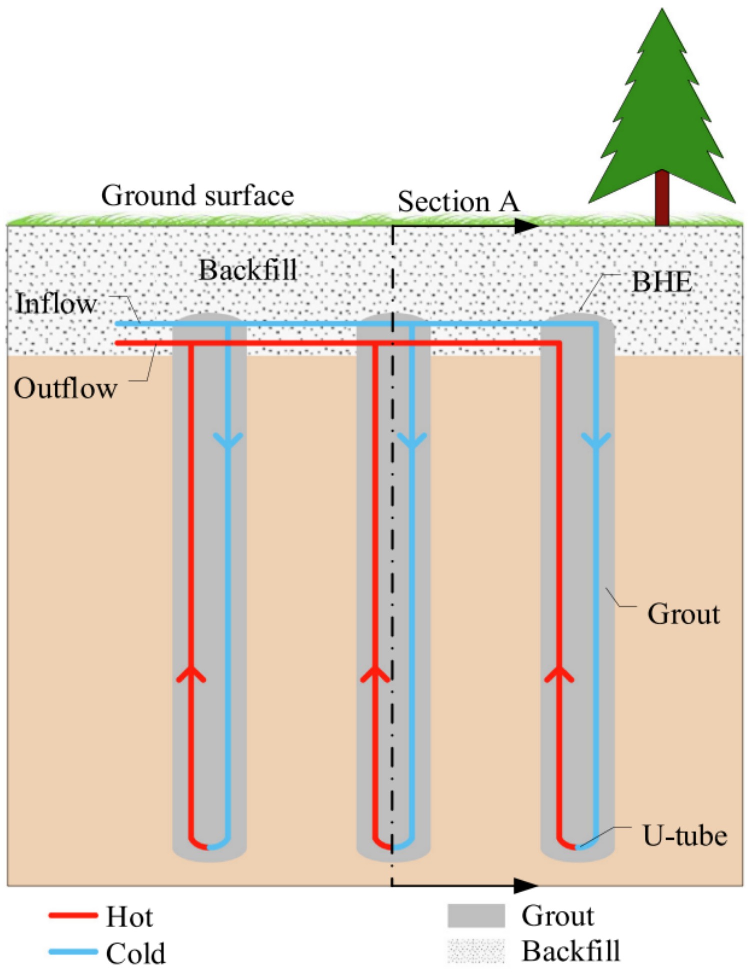


Figure 2.8.: Schematic of borehole field with three borehole heat exchangers, taken from [39]

field. These fields comprise a number of vertically aligned BHEs, each of which consists of a hole drilled in the ground at a depth of 30 to 100 meters [37]. Further, every hole typically contains one, or sometimes more, U-tubes [38]. The resulting space between the tubes and the borehole wall is usually backfilled or grouted. The heat-carrying liquid inside the tubes is typically water (with antifreeze additives) that circulates between the ground and the heat pump [39]. A schematic of such a vertical borehole field is given in Figure 2.8.

Different BHE Designs

The most common type of borehole heat exchangers is given by U-pipe BHEs, as presented in Figure 2.8. For this design, both the upward and downward flow yield contributions to the heat exchange between the water in the tubes and the surrounding ground.

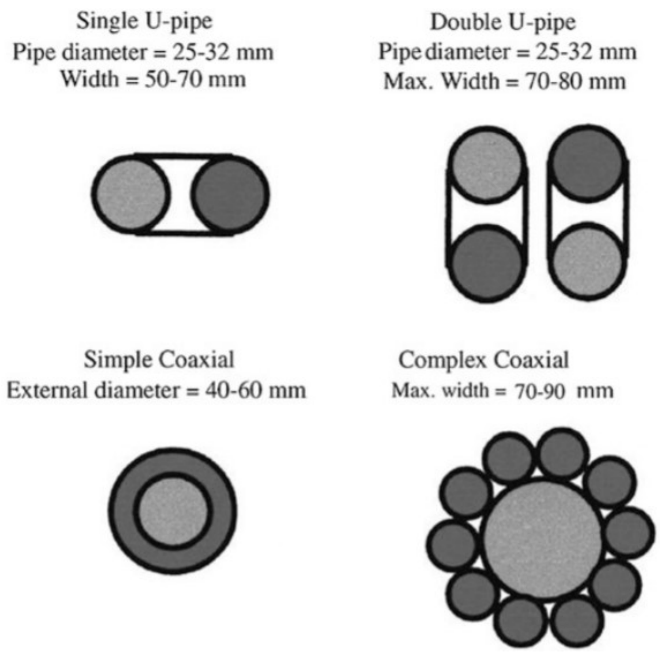


Figure 2.9.: Schematic of common BHE designs, taken from [40]

There are both single U-pipe as well as double U-pipe BHE types, with the single version being more common [40].
Apart from U-pies, also concentric pipe designs exist where one straight pipe is located in a second, bigger pipe. This BHE design features heat exchange only from either upward or downward pipe. Often, the inner pipe is insulated to avoid disturbances. Another common BHE type is represented by coaxial pipes, which can be designed with or without outer tube, thereby featuring either closed or open flow circuits. A schematic of these different BHE designs is given in Fig. Figure 2.9 [40].

Thermodynamic Modeling of Ground Heat Transfer

In order to make predictions on the thermal output of the BHEs, several assumptions need to be made, to simplify the complex heat transfer phenomena. In essence, the heat flow from the ground can be calculated by multiplying the geothermal gradient with the thermal conductivity of the ground. The thermal conductivity highly depends on the type of rock material in the ground; quartz-rich rocks, for instance, have a high thermal conductivity, whereas rocks rich in clay have low thermal conductivity.
The overall heat transfer in a BHE is composed of two parts. One of them is the heat conduction outside the borehole, i.e. the transfer of the heat from the ground to the borehole. Several models have been developed to simulate this thermodynamic process, among the most prominent ones is Eskilson's Model [41]. This model is mainly concerned with relating the heat extraction rate with the temperature of the circulating fluid in the U-tube. Eskilson starts by analyzing the thermal process of a single borehole, before

investigating the interaction between the several boreholes of the borehole field. To describe these interactions and the overall thermal performance of the borehole field, he introduces the so-called g-functions, which are dimensionless time response factors. Eskilson developed a numerical model for variable configuration of boreholes and used superposition to derive the thermal influence of multiple boreholes. In addition, an analytical approach is used to solve for the heat transfer process along the borehole. The assumptions made by Eskilson's model include: annual mean air temperature equals the ground surface temperature, the ground material is homogeneous, the top section of the borehole is thermally insulated, and the borehole temperature is constant along the borehole depth. Under these assumptions, the heat conduction equation given in Equation 2.22 for a single borehole can be derived by means of the Laplacian of the scalar temperature function $T(x, y, z, t)$

$$\Delta T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \tag{2.30}$$

as

$$\frac{1}{a}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial T^2}{\partial z^2}, \tag{2.31}$$

where a corresponds to the thermal conductivity, T is the temperature, t gives the time, and x, y, z are cartesian coordinates. In cylindrical coordinates, this corresponds to the form:

$$\frac{1}{a}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2} \tag{2.32}$$

with the cylindrical coordinates in radial and vertical direction r and z. The boundary conditions are given as:

$$\begin{aligned} T(r, 0, t) &= T_o \\ T(r, z, 0) &= T_o \\ T(r, \infty, t) &= T_o \\ \frac{\partial T}{\partial r}(0, z, t) &= 0, \end{aligned} \tag{2.33}$$

where T_o is the ground surface temperature. When modeling heat extraction or rejection as step function $Q(t)$, the temperature response can be determined using a single step pulse, with the final solution obtained by summing over the corresponding step pulse responses over time. A schematic of this calculation is given in Figure 2.10. By converting the step-wise temperature responses to dimensionless response factors (g-functions), the borehole temperature $T_b(t)$ can be calculated as [40]:

$$T_b(t) = T_o + \sum_{i=1}^{n} \frac{Q_i - Q_{i-1}}{2\pi k} g\Big(\frac{t_n - t_{i-1}}{t_s}, \frac{r_b}{H}\Big), \tag{2.34}$$

where r_b is the borehole radius, Q_i is the thermal load extracted or injected at time step i, k is the thermal conductivity of the surrounding material and H is the active

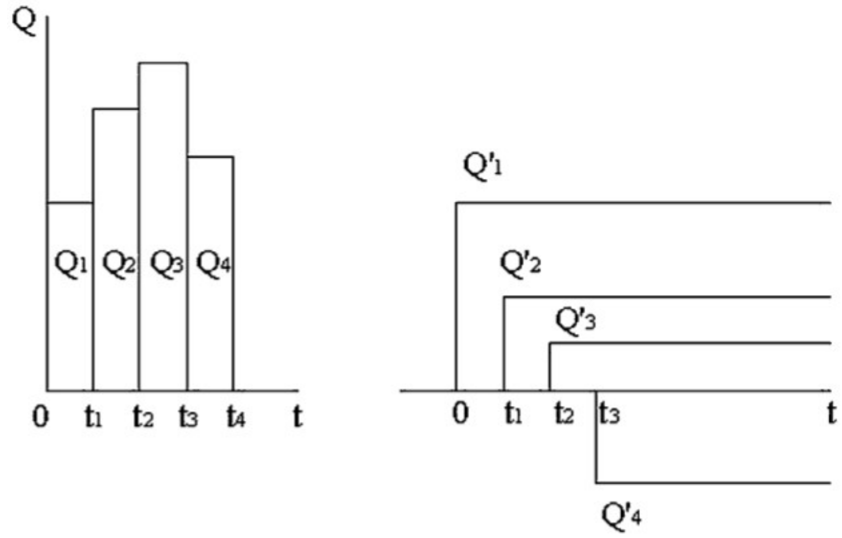


Figure 2.10.: Schematic of heat step functions and step pulses, taken from [40]

borehole length, i.e. the vertical length of the borehole that is effectively involved in the heat exchange process. A basic time-scale $\frac{5r_b^2}{\alpha}$ was introduced, below which heat transfer rates are considered very imprecise.

The second part, contributing to the overall heat transfer in a BHE is given by the heat transfer inside the borehole. The goal of modeling this process is to determine the temperature of the circulating fluid at the inlet and outlet of the borehole, in correspondence with the borehole wall temperature, the thermal resistance, and the heat flow. A commonly used approach for this aim is the two-dimensional thermal resistance model derived by Hellstöm [42]. In this model, the fluid temperature is obtained by summing over the two individual temperature responses to the heat fluxes per unit length, q_1, q_2. By assuming constant borehole wall temperature T_b along the depth of the borehole, the fluid temperatures T_{f1}, T_{f2} for upward and downward pipes in a U-tube can be calculated as:

$$\begin{aligned} T_{f1} - T_b &= R_{11}q_1 + R_{12}q_2 \\ T_{f2} - T_b &= R_{12}q_1 + R_{22}q_2, \end{aligned} \tag{2.35}$$

with R_{11} and R_{22} being the thermal resistances between the fluid and the borehole wall, and R_{12} being the resistance between the two pipes of the U-tube. The thermal resistances are calculated as the inverse of the thermal conductivities. Since the model incorporates some simplifications, such as neglecting heat transmission on the axial flow of the circulating fluid, there is in fact no distinction between upward and downward pipe. Therefore, one can set $T_{f1} = T_{f2} = T_f$ and $q_1 = q_2$, which leads to the thermal resistance between borehole and fluid of

$$R_{b2} = \frac{R_{11} + R_{12}}{2}. \tag{2.36}$$

With this being given, the temperature of the fluid at the borehole outlet can be obtained. It should be noted, however, that actually the temperature of the fluid in the upward pipe differs from the temperature in the downward pipe, thus, introducing an inaccuracy to the model [40].

2.3. Quantum Chemistry

Quantum chemistry is an interdisciplinary field that deals with the application of quantum mechanics to problems in chemistry. It aims to investigate the electronic structure of atoms and molecules in order to provide insights into their properties such as bonding, reactivity, and interaction. Through the emergence of advanced computational possibilities and the development of new methods for molecular calculations such as density functional theory (DFT), quantum chemistry not only enhances our theoretical knowledge but also paves the way for advancements in various applications, ranging from materials science and nanotechnology to renewable energy solutions and pharmaceutical research [43].

2.3.1. Schrödinger Equation and Born-Oppenheimer Approximation

The total (non-relativistic) time-independent Schrödinger equation for a many-particle system is given as:

$$\begin{aligned} \mathbf{H}_{\text{tot}}\Psi_{tot} &= E\Psi_{tot}, \\ \mathbf{H}_{\text{tot}} &= -\sum_i \frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_a \frac{\hbar^2}{2m_a}\nabla_a^2 - \sum_{a,i}\frac{Z_a \mathrm{e}^2}{4\pi\epsilon_0|\mathbf{r}_i - \mathbf{R}_a|} \\ &\quad + \sum_{i<j}\frac{\mathrm{e}^2}{4\pi\epsilon_0|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{a<b}\frac{Z_a Z_b \mathrm{e}^2}{4\pi\epsilon_0|\mathbf{R}_a - \mathbf{R}_b|}, \end{aligned} \tag{2.37}$$

where electrons are denoted with subscript i and nuclei with subscript a, and Z_a is the number of protons in nucleus a. The total Hamiltonian can be rewritten in short form as:

$$\mathbf{H}_{\text{tot}} = \mathbf{T}_e + \mathbf{T}_n + \mathbf{V}_{\text{ne}} + \mathbf{V}_{\text{ee}} + \mathbf{V}_{\text{nn}}. \tag{2.38}$$

In the Born-Oppenheimer approximation, the positions of the nuclei are assumed to be fixed. This is a reasonable approximation since the nuclei are much heavier than the electrons, thus, the movement of the electrons happens on another time scale than the movement of the nuclei. Thus, the electronic part of the Schrödinger equation can be expressed as:

$$\mathbf{H}_{elec}\Phi_m(\mathbf{r},\mathbf{R}) = (T_e + V(\mathbf{r},\mathbf{R}))\Phi_m(\mathbf{r},\mathbf{R}) = W_m\Phi_m(\mathbf{r},\mathbf{R}), \tag{2.39}$$

where $V(\mathbf{r},\mathbf{R})$ is the potential energy operator and W_m is the energy eigenvalue of the electronic wave function $\Phi_m(\mathbf{r},\mathbf{R})$. The electronic coordinates are denoted as $\mathbf{r}$ and the nuclear positions as R. The total solution of the Schrödinger equation can be expanded in electronic coordinates by

$$\Psi_{\text{tot}}(\mathbf{R},\mathbf{r}) = \sum_{m=0}^{\infty}\chi_m(\mathbf{R})\Phi_m(\mathbf{r},\mathbf{R}), \tag{2.40}$$

where $\chi_m(\mathbf{R})$ represents the nuclear wave function. Inserting the expression into the Schrödinger equation yields

$$\Big(\mathbf{T}_e + \mathbf{T}_n + \mathbf{V}(\mathbf{r},\mathbf{R})\Big)\sum_{m=0}^{\infty}\chi_m(\mathbf{R})\Phi_m(\mathbf{r},\mathbf{R}) = E\sum_{m=0}^{\infty}\chi_m(\mathbf{R})\Phi_m(\mathbf{r},\mathbf{R}) \tag{2.41}$$

After some algebra (for more in-depth explanation see [44]), we obtain:

$$\Big[\mathbf{T}_n + W_l(\mathbf{R}) - E\Big]\chi_l(\mathbf{R}) = \sum_m \Lambda_{lm}\chi_m(\mathbf{R}), \tag{2.42}$$

where Λ_{lm} represents the coupling term between electrons and nuclei. Since the Born-Oppenheimer approximation assumes that nuclear and electronic motion are decoupled, we set $\Lambda_{lm} = 0$ and rewrite the nuclear part of the Schrödinger equation as:

$$\Big[\mathbf{T}_n + W_l(\mathbf{R}) - E\Big]\chi_n(\mathbf{R}) = 0. \tag{2.43}$$

Therefore, the total wave function can be expressed as product of nuclear and electronic wave functions:

$$\Psi_{lv}(\mathbf{r}, \mathbf{R}) = \Phi_l(\mathbf{r}, \mathbf{R})\chi_v(\mathbf{R}). \tag{2.44}$$

A requirement for the Born-Oppenheimer Approximation to be valid is:

$$\frac{|\langle\chi_{ls}|\,\Lambda_{lm}\,|\chi_{ml'}\rangle|}{|E_{ls} - E_{ms'}|} \ll 1. \tag{2.45}$$

Within the Born-Oppenheimer approximation, the nuclei can be considered to move in a potential energy surface (PES), created by the electronic configuration of the system. The energy of the many-electron system can be calculated as:

$$E = \frac{\langle\Psi|\,\mathbf{H}_{elec}\,|\Psi\rangle}{\langle\Psi|\Psi\rangle}. \tag{2.46}$$

The structure of a molecule, with respect to its bond lengths and bond angles is given by the structure that minimizes this expression [45, 44, 46].

2.3.2. Geometry Optimization

A central component of energy calculation is the initial geometry optimization, i.e. search of the structure that minimizes the energy of the investigated system. For this purpose, several methods of low-energy structure search for nano-particles have been developed. A concise overview is given in [47]. The standard method for geometry optimization in molecules is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [48], which is based on a pseudo-Newton method. For structure optimization on complicated PES with numerous local minima, however, the Basin-hopping algorithm is a more suitable approach. In this method, the shape of the PES is altered, while local minima are retained exactly in the transformed PES. The transformation is conducted by calculating local minima for every point in the configuration space:

$$\tilde{E}(\mathbf{X}) = \min\{E(\mathbf{X})\}. \tag{2.47}$$

This procedure would correspond to transformation into a staircase function for a 2D representation. Then, for the optimization, the sampling is carried out through move-upon atom coordinates, with a starting configuration S and a destination configuration

D. Each individual *move* can be either refused or accepted, in accordance with the standard Metropolis algorithm:

$$p = \begin{cases} 1, & \text{if } \tilde{E}_D \leq \tilde{E}_S \text{ odd} \\ e^{-(\tilde{E}_D - \tilde{E}_S)/K_B T}, & \text{if } \tilde{E}_D > \tilde{E}_S \end{cases}, \tag{2.48}$$

with p representing the probability that the *move* is accepted and T representing the fictitious temperature of the system. A more in-depth discussion, and explanation of the different types of *moves* is presented in [47].

2.3.3. DFT Basics

Hohenberg and Kohn [49] proved in 1964 that the energy of the electronic ground state is completely determined by the electron density ρ. This means that the electron density of a system uniquely defines its energy. An intuitive proof for this was formulated by E.B. Wilson, who claimed that (i) the integral over the electron density yields the number of electrons, (ii) the position of the nuclei are given by the cusps in the electron density, and (iii) the heights of the cusps determine the nuclear charges [44]. The basic idea of DFT is thus to express the energy of an electronic system as a functional of the electron density, $E[\rho]$. This bears the advantage of DFT that the energy expression is only dependent on three spatial coordinates. i.e. independent of the number of electrons. In contrast, wave-function-based approaches for an N-electron system depend on $4N$ variables, three spatial and one spin coordinate for each electron [44].

Orbital-Free DFT

Early attempts of DFT used orbital-free expressions and divided the total energy functional of the electronic system into three parts: attractive potential energy between nuclei and electrons $E_{\text{ne}}[\rho]$, repulsive potential energy between electrons $E_{\text{ee}}[\rho]$, and the kinetic energy of the electrons $T[\rho]$. The electron-electron interaction term can be further divided into Coulomb part $J[\rho]$ and exchange part $K[\rho]$. The functionals for the Coulomb part and the nuclei-electron interaction can be straightforwardly defined as:

$$E_{\text{ne}}[\rho] = -\sum_{a}^{N_{nuclei}} \int \frac{Z_a(\mathbf{R}_a)\rho(\mathbf{r})}{|\mathbf{R}_a - \mathbf{r}|} d\mathbf{r}, \tag{2.49}$$

$$J[\rho] = \frac{1}{2} \iint \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r} d\mathbf{r}'. \tag{2.50}$$

However, the remaining functionals can not be defined as easily. Early attempts were based on the assumption of the uniform electron gas, which yields somewhat correct results for metals, but deviates strongly from the behavior in molecules. The corresponding functionals are given as

$$\begin{aligned} T_{TF}[\rho] &= C_F \int \rho^{5/3}(\mathbf{r})d\mathbf{r} \\ K_D[\rho] &= -C_x \int \rho^{4/3}(\mathbf{r})d\mathbf{r} \\ C_F &= \frac{3}{10}(3\pi^2)^{2/3} \\ C_x &= \frac{3}{4}(\frac{3}{\pi})^{1/3}, \end{aligned} \tag{2.51}$$

with the subscripts $_{TF}$ denoting the Thomas-Fermi-theory and $_D$ Dirac's theory [44].

Kohn-Sham DFT

In Kohn-Sham DFT, orbitals are reintroduced to overcome the poor representation of the kinetic energy term in orbital-free approaches. The idea is to split the most problematic functional of the kinetic energy into one part that can be calculated exactly and another part representing a small correction term. Due to the reintroduction of orbitals, the complexity is increased to $3N$ variables and electron-electron interaction re-emerges. The Hamiltonian is assumed to have the form:

$$\mathbf{H}_\lambda = \mathbf{T} + \mathbf{V}_{\text{ext}}(\lambda) + \lambda \mathbf{V}_{\text{ee}}, \tag{2.52}$$

with $0 \leq \lambda \leq 1$ defining the influence of the electron-electron interaction V_{ee}. Thus, for $\lambda = 1$, the external potential V_{ext} equals V_{ne}. For $\lambda = 0$, the system is assumed to have no electron-electron interaction, so that the exact solution of the Schrödinger equation is given by a Slater determinant of molecular orbitals. For values of λ between 0 and 1, the external potential $V_{ext}(\lambda)$ needs to be adjusted in a way that the electron density remains the same. For the $\lambda = 0$ case of non-interacting electrons, the exact kinetic energy can be expressed as:

$$T_S = \sum_{i=1}^{N_{elec}} \langle \phi_i | -\frac{1}{2}\nabla^2 | \varphi_i \rangle, \tag{2.53}$$

with ϕ_i denoting the i-th molecular orbital. This expression for the kinetic energy, which actually only holds for non-interacting electrons, is now also used for $\lambda > 0$, i.e. for real systems including electron-electron interactions. Therefore, the expression for the kinetic energy in Kohn-Sham DFT is again just an approximation, but it represents a significant enhancement over the Thomas-Fermi formula used in Equation 2.51.

Using the equation for the kinetic energy in Equation 2.53 is justified by the fact that exact kinetic energy can also be obtained from the *natural orbitals* (NO) that arise from the exact density matrix:

$$T[\rho_{\text{exact}}] = \sum_{i=1}^{\infty} \langle \phi_i^{\text{NO}} | -\frac{1}{2}\nabla^2 | \phi_i^{\text{NO}} \rangle$$
$$\rho_{\text{exact}} = \sum_{i=1}^{\infty} n_i |\phi_j^{\text{NO}}|^2 \tag{2.54}$$
$$N_{\text{elec}} = \sum_{i=1}^{\infty} n_i,$$

where the n_i represents the number of electrons in the spin orbitals (between 0 and). To express the exact electron density, infinitely many NO are needed, where the first N_{elec} orbitals will have $n_i \approx 1$ and the following ones $n_i \approx 0$. Hence, the approximated electron density can be expressed by means of a set of auxiliary one-electron orbitals [44]:

$$\rho_{\text{approx}} = \sum_{i=1}^{N_{\text{elec}}} |\phi_i|^2 \tag{2.55}$$

Under the assumption of non-interacting electrons to justify the use of Equation 2.53, the total DFT energy can then be expressed as:

$$E_{\text{DFT}}[\rho] = T_S[\rho] + E_{\text{ne}}[\rho] + J[\rho] + E_{\text{xc}}[\rho], \tag{2.56}$$

which automatically determines the expression of the exchange-correlation functional E_{xc} as some type of correction term, by setting E_{DFT} equal to the exact energy, yielding

$$E_{\text{xc}}[\rho] = \underbrace{(T[\rho] - T_S[\rho])}_{\text{kinetic correlation energy}} + \underbrace{(E_{\text{ee}}[\rho] - J[\rho])}_{\text{exchange-correlation energy}} . \tag{2.57}$$

Often, the exchange and correlation energies are also written in terms of the energy per particle:

$$E_{\text{xc}}[\rho] = E_{\text{x}}[\rho] + E_c[\rho] = \int \rho(r)\epsilon_x[\rho(r)]dr + \int \rho(r)\epsilon_c[\rho(r)]dr. \tag{2.58}$$

The only part that needs to be derived in Kohn-Sham DFT is an approximation for the exchange-correlation energy E_{xc}, while for orbital-free DFT expressions for kinetic energy, exchange- and correlation contributions need to be determined [44].

Kohn and Sham inferred that a system of N interacting electrons can be expressed by means of N non-interacting electrons moving in an external potential:

$$\left\{-\frac{\hbar^2\nabla^2}{2m} + v_{\text{ext}}(\mathbf{r})\right\}\phi_i(\mathbf{r}) = \epsilon_i\phi_i(\mathbf{r}), \tag{2.59}$$

where ϕ_i are the one-electron orbitals (also known as Kohn-Sham orbitals) and ϵ_i are the corresponding eigenvalues, i.e. the associated energies of these one-electron orbitals. Since the density $\rho(\mathbf{r})$ is required for the calculation of the external potential v_{ext}, and the density, in turn, depends on the external potential, this potential needs to be found in a *self-consistent* manner. That is, an initial guess for $\rho(\mathbf{r})$ is made to construct the potential, thereafter, the resulting Kohn-Sham equations are solved to find an updated $\rho(\mathbf{r})$, which is then used to calculate the new potential. This procedure is iterated until convergence is reached. This self-consistent loop is the key element in any DFT code [50].

2.3.4. DFT Improvements / Exchange-Correlation Functionals

Various DFT methods have been developed to date, typically differing only in the choice of the functional expressing the exchange-correlation energy. Since these exchange-correlation functionals are mostly empirical, there is no theoretically sound order of the different methods. However, a heuristic characterization, considering the fundamental variables used, has been established and is known as *Jacob's ladder*. For each step up the ladder, an improvement in the accuracy is at least hoped to be achieved.
The first step on Jacob's ladder is given by the *Local Density Approximation.* This method treats the density locally as a uniform electron gas, such that the density is a slowly varying function. The exchange energy is then given by:

$$\begin{aligned} E_{\mathrm{x}}^{\mathrm{LDA}}[\rho] &= -C_x \int \rho^{4/3}(\mathbf{r}) d\mathbf{r}, \\ \epsilon_{\mathrm{x}}^{\mathrm{LDA}} &= -C_{\mathrm{x}} \rho^{1/3} \end{aligned} \tag{2.60}$$

which provides an upper bound to the exchange-correlation functional [44].

$$E_{\mathrm{x}}[\rho] \geq E_{\mathrm{xc}}[\rho] \geq 2.273 E_x^{\mathrm{LDA}}[\rho]. \tag{2.61}$$

The correlation energy part of a uniform electron gas does not have a simple analytical form but is typically obtained for with Monte Carlo methods.
On the next step of Jacob's ladder *generalized gradient approximation* (GGA) methods. These methods include not only the density ρ, but also its gradient $\nabla\rho$ as variable for the formulation of the functional. Among the most popular GGA exchange functionals for the exchange energy is:

$$\begin{aligned} \epsilon_{\mathrm{x}}^{\mathrm{B88}} &= \epsilon_{\mathrm{x}}^{\mathrm{LDA}} + \nabla\epsilon_{\mathrm{x}}^{\mathrm{B88}} \\ \nabla\epsilon_{\mathrm{x}}^{\mathrm{B88}} &= -\beta\rho^{1/3} \frac{x^2}{1 + 6\beta x \sinh^{-1}(x)} \\ x &= \frac{|\nabla\rho|}{\rho^{4/3}}, \end{aligned} \tag{2.62}$$

where β is a parameter, defined by fitting the equations to known data. Various GGA functionals have been derived for the correlation energy, with one of the most common functionals represented by LYP [51]:

$$\begin{aligned}\epsilon_c^{\mathrm{LYP}} &= -4a\frac{\rho_\alpha\rho_\beta}{\rho^2(1+d\rho^{-1/3})} - \frac{\rho_\alpha\rho_\beta}{18\,ab\,\omega}\Bigg\{144(2^{2/3})C_F(\rho_\alpha^{8/3}+\rho_\beta^{8/3}) + (47-7\delta)|\nabla\rho|^2 \\ &\quad - (45-\delta)(|\nabla\rho_\alpha|^2+|\nabla\rho_\beta|^2) + 2\rho^{-1}(11-\delta)(\rho_\alpha|\nabla\rho_\alpha|^2+\rho_\beta|\nabla\rho_\beta|^2) \\ &\quad + \frac{2}{3}\rho^2(|\nabla\rho_\alpha|^2+|\nabla\rho_\beta|^2-|\nabla\rho|^2) - (\rho_\alpha^2|\nabla\rho_\beta|^2+\rho_\beta^2|\nabla\rho_\alpha|^2)\Bigg\},\end{aligned} \tag{2.63}$$

with

$$\omega = \frac{e^{-cp^{-1/3}}}{\rho^{14/3}(1+d\rho^{-1/3})} \tag{2.64}$$

and

$$\delta = c\rho^{-1/3} + \frac{d\rho^{-1/3}}{1+d\rho^{-1/3}}. \tag{2.65}$$

The parameters a, b, c, d are fitted to the data for a helium atom. By combining the LYP correlation functional with the B88 exchange functional, BLYP acronyms are obtained. Perdew, Burker, and Ernzerhof have introduced the PBE functional [52], which can be considered a refinement, based on removing spurious oscillations in the first-order Taylor-like expansion of the exchange-correlation energy density around the uniform electron gas model. The exchange part of the functional is given as:

$$\begin{aligned}\epsilon_x^{\mathrm{PBE}} &= \epsilon_x^{\mathrm{LDA}}F(x) \\ F(x))1 + a &- \frac{a}{1+bx^2} \\ x &= \frac{|\nabla\rho|}{\rho^{4/3}}.\end{aligned} \tag{2.66}$$

The correlation term is analogously represented as an enhancement factor added to the LSDA functional, with t being linked to x through yet another spin-polarization function.

$$\begin{aligned}\epsilon_c^{\mathrm{PBE}} &= \epsilon_c^{\mathrm{LDA}} + H(t) \\ H(t) = cf_3^3 \ln &\left[1 + dt^2\left(\frac{1+At^2}{1+At^2+A^2t^4}\right)\right] \\ A &= d\left[\exp\left(-\frac{\epsilon_c^{\mathrm{LDA}}}{cf_3^3}\right) - 1\right]^{-1} \\ f_3(\zeta) &= \frac{1}{2}\left[(1+\zeta)^{2/3} + (1-\zeta)^{2/3}\right] \\ t &= \left[2(3\pi^2)^{1/3}f_3\right]^{-1}x.\end{aligned} \tag{2.67}$$

For this functional, the parameters a, b, c, d are not fitted to data, but derived from certain conditions [44].

2.3.5. Basis Sets, Plane-Waves, Pseudopotentials

Basis sets are used to express unknown functions, such as molecular orbitals or electronic wavefunctions. If the expansion of the unknown functions is carried out with a *complete* basis set, this yields an exact representation of the unknown function. However, a complete basis set requires an infinite number of functions and is therefore intractable for actual calculations. The two most commonly used types of basis functions in electronic structure theory are *Slater Type Orbitals* (STOs)

$$\chi_{\zeta,n,l,m}(r,\theta,\phi) = NY_{l,m}(\theta,\phi)r^{n-1}e^{-\zeta r} \tag{2.68}$$

where N is a normalization constant, $Y_{l,m}$ are the spherical harmonics; and *Gaussian Type Orbitals* (GTOs):

$$\chi_{\zeta,n,l,m}(r,\theta,\phi) = NY_{l,m}(\theta,\phi)r^{2n-2-l}e^{-\xi r^2}. \tag{2.69}$$

Instead of modeling atomic orbitals (GTOs or STOs) with basis functions, plane wave basis functions aim to directly model the full system. For instance, to model extended systems such as an infinitely repeated unit cell with periodic boundary conditions, it is a more natural choice to employ functions with infinite range. In metals, the behavior of the valence electrons closely resembles the behavior of free electrons. Therefore, solutions to the Schrödinger equations for a free electron can be used as ansatz for basis functions:

$$\begin{aligned}\phi(x) &= Ae^{ikx} + Be^{-ikx}\\ \phi(x) &= A\cos(kx) + B\sin(kx),\end{aligned} \tag{2.70}$$

with the energy

$$E = \frac{1}{2}k^2. \tag{2.71}$$

In infinite systems, the energy difference between consecutive orbitals vanishes, thereby leading to a blending of the discrete molecular orbitals into *bands*. Electrons residing in these bands are described using orbitals that are expressed in terms of a plane-wave basis, which can be represented as:

$$\chi_k(\mathbf{r}) = e^{i\mathbf{k}\cdot\mathbf{r}}, \tag{2.72}$$

where k is the wave vector that takes the same role as ξ in a GTO. The allowed values of k are defined by the unit cell translational vector $\mathbf{t}$ with $\mathbf{k}\cdot\mathbf{t} = 2\pi m$ and m being a positive integer [44].

From a chemical bonding perspective, core electrons do not significantly contribute to the interactions in a system. However, it is vital to include a large enough number of basis functions in order to describe their orbitals sufficiently well, as otherwise, the orbitals of the valence electrons will be inaccurately represented due to a lack of electron-electron repulsion. This problem can be addressed by modeling the core electrons with *pseudopotentials* (also called *effective core potentials*). These pseudopotentials are designed by first, creating a reasonable all-electron wave function (e.g. with DFT calculations), then replacing the orbitals of the valence electrons by nodeless pseudo-orbitals. These orbitals behave correctly in the outer region, but do not capture the true behavior near the core due to the missing nodal structure. Thus, as a next step, the core electrons are replaced by parameterized analytical functions (e.g. Gaussian functions) of the nuclear-electron distance. Finally, the parameters of the created potentials are fitted in a way that the wave functions found as solutions of the Schrödinger equation yield pseudo-orbitals that match the all-electron valence orbitals. Traditionally, molecular systems were modeled using Gauss-type basis sets, whereas extended periodic systems were described by plane waves. Consequently, also the corresponding pseudopotentials vary for these systems, i.e. when Gaussian functions are used for the description of valence electrons, they are naturally also used for the description of the core electrons. Apart from replacing all electrons except for the valence electrons by pseudopotentials, one can also include orbitals of the next-lower orbital under the valence orbital for explicit treatment. For example, for silver (Ag) atom, one may choose a 'large-core' or a 'small-core' pseudopotential, which are given as follows (remaining electrons are bold and replaced electrons italic) [44]:

- *Large-core* pseudopotential (11 electrons explicitly considered):
  (1s)^2 (2s)^2 (2p)^6 (3s)^2 (3p)^6 (4s)^2 (3d)^{10} (4p)^6 **(4d)^{10} (5s)^1**
- *Small-core* pseudopotential (19 electrons explicitly considered):
  (1s)^2 (2s)^2 (2p)^6 (3s)^2 (3p)^6 **(4s)^2** *(3d)^{10}* **(4p)^6 (4d)^{10} (5s)^1**
- *All-electron* potential (47 electrons explicitly considered):
  (1s)^2 (2s)^2(2p)^6 (3s)^2 (3p)^6 (4s)^2 (3d)^{10} (4p)^6 (4d)^{10} (5s)^1

This yields the orbital shapes represented in Figure 2.11

2.3.6. Computational Tools for Electronic Structure Calculations

Computational tools play a pivotal role in calculating the electronic structure and chemical properties of materials. Therefore, a number of software packages has been developed for computational quantum chemistry. The core working principle that they share is to solve the Schrödinger equation for electronic systems, using different methods and approximations. Within this work, we will use *Quantum Espresso* for the DFT calculations to obtain the total energy $E(\mathbf{x})$ of the given structure and the gradient $\nabla_{\mathbf{x}} E$ of it. *"Quantum Espresso is an integrated suite of Open-Source computer codes for electronic-structure calculations and materials modeling at the nanoscale. It is based on density-functional theory, plane waves, and pseudopotentials"* [53]. It is, therefore, well suited for calculations of clusters with periodic boundary conditions and bonding properties typically present in nanoclusters.

Assuming that the DFT calculations are carried out for solids exhibiting discrete translational invariance, the Bloch theorem can be used, which leads to the advantages that the Hamiltonian is diagonal in $\mathbf{k}$ and the calculations can be restricted to one single

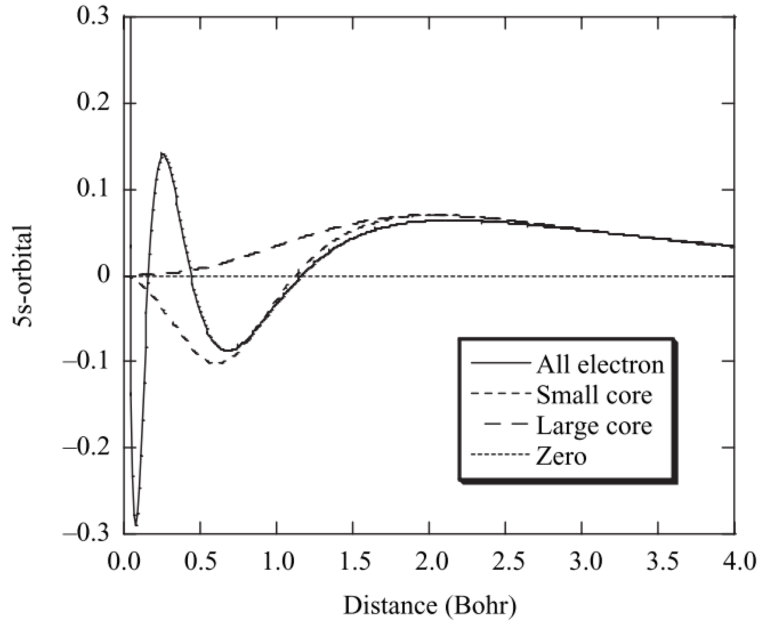


Figure 2.11.: 5s-orbital of Ag with all-electron, small-core, or large-corre effective core potential. Image taken from [44].

unit cell. With the Bloch basis, using $u_{\mathbf{k}}(\mathbf{r})$ lattice-periodic functions and $\mathbf{k}$ as the wave vector of the first Brillouin zone, Equation 2.59 can be rewritten as:

$$\Big(-\frac{\hbar^2}{2m}(\frac{1}{i}\Delta + \mathbf{k})^2 + v_{\text{ext}}(\mathbf{r})\Big)u_{\mathbf{k}}(\mathbf{r}) = \epsilon_{\mathbf{k}}u_{\mathbf{k}}(\mathbf{r}), \tag{2.73}$$

which defines the dispersion relation $\epsilon_{\mathbf{k}}$. From this equation, it is evident that we simply need to solve a single-particle equation for various values of $\mathbf{k}$, since the Hamiltonian is diagonal in $\mathbf{k}$. I.e., for each $\mathbf{k}$, we obtain a set of $\epsilon_i(\mathbf{k})$, which corresponds to the dispersion relation. Rather than directly solving the second order differential equation Equation 2.73, the wave function $u_{\mathbf{k}}(\mathbf{r})$ can be expanded in terms of a linear combination of basis functions, which yields a linear system of equations. The number of basis functions (plane waves) included in the expansion is determined in *Quantum Espresso* by a cutoff energy. This ideal cutoff depends on the element(s) and the size of the unit cell and can be found by running calculations with different cutoff energies to check when the resulting total energy converges. By using pseudopotentials, the number of necessary plane waves in the basis set can be significantly reduced [50].

3. PINNs for Borehole Heat Exchangers

The aim of this work is to create physics-informed machine learning models, i.e. a physics-informed LSTM (PI-LSTM) that enables the prediction of the temperature output of a borehole heat exchanger, with a prediction horizon of 1 or 2 weeks, given a look-back window of 7 - 14 days. Most of the BHE models introduced in previous studies [9, 10, 38] are based on simulations of long-term responses. The few models concerned with the prediction of short-term behavior typically rely on finite element methods, which go along with considerable computational efforts. In our study, we try to establish a model that is able to accurately predict short- and mid-range behavior of the system at manageable computational costs.

This chapter explores the equations governing the heat transfer process in Section 3.1.2, before delving into the practical implementation of a BHE system in a real-world use-case. Its corresponding data will be examined in Section 3.1, followed by sections about the subsequent steps: Preprocessing, establishment of a baseline, and implementation of the proposed physics-informed model. After the presentation of the results in Section 3.4, the chapter will conclude with a concise discussion of our findings.

3.1. Experimental Setup

The most important requirement for the implementation of our proposed PI-LSTM is a realistic dataset, since the aim of this work is to establish a solid prediction approach for real-world operating systems. In addition, a large number of data points (i.e. a long enough time span) is necessary to enable concise validation for both heating (winter) and cooling (summer) periods. The following sections will introduce the employed dataset and explain the conducted preprocessing steps to obtain a suitable training set.

3.1.1. Dataset Description

We opted for a dataset of a ground-source heat pump system located at an office building at the Universitat Politècnica de València, which was extensively studied by Ruiz-Calvo et al. [9]. The dataset spans an operation period of 11 years and has previously been used for validation of several purely theoretical models [10, 38, 54].
The operation of the system started in February 2005 and ought to provide both heating and cooling to several rooms with a total area of roughly 250 m^2. A schematic of the GSHP installation is given in Figure 3.1. The whole system mainly consists of two hydraulic loops: an internal (building) loop connecting the individual fan coil units distributed over the rooms; and an external (ground) loop coupling the GHE to the heat pump. An electronic controller determines the operational state of the heat pump, by

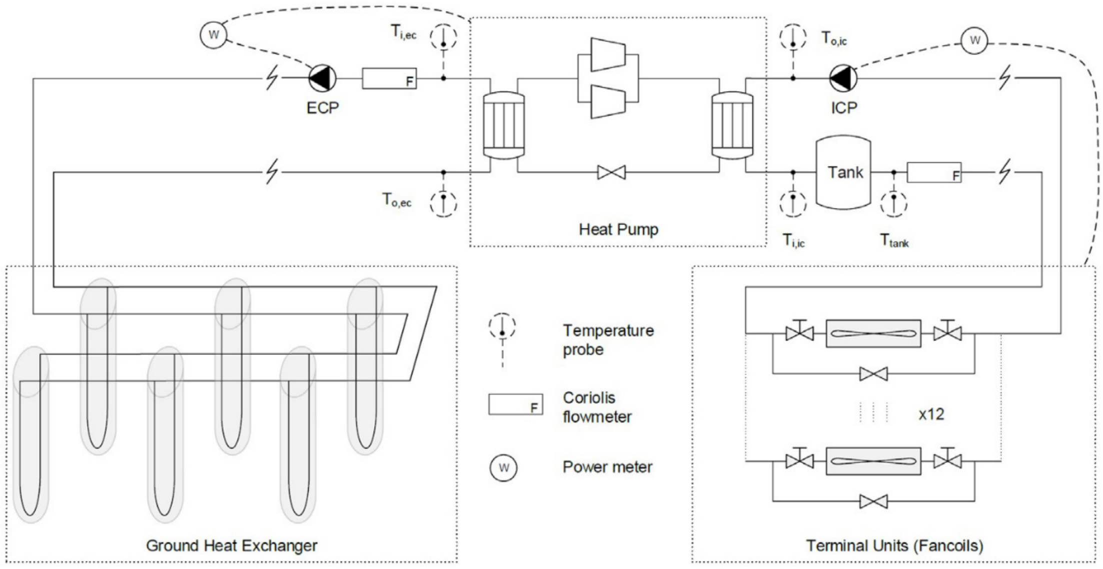


Figure 3.1.: Schematic of the GSHP of the employed dataset from Ruiz-Calvo et al. [9]

switching its compressor on or off, depending on the temperature of the internal loop's return water. The specific type of the GSHP in usage is called a "reversible water-to-water heat pump". Initially, it was constructed as a single-stage (on/off) heat pump using propane R290 as refrigerant, featuring a nominal cooling capacity of 14.7 kW and a nominal heating capacity of 17 kW. In May 2011 the initial heat pump was replaced by a new water-to-water reversible heat pump, using the refrigerant R410A. In addition, by utilizing water-reversing valves the new heat pump's design allows it to operate reversibly on both the refrigerant and the secondary fluid loops. This feature enables continuous operation in counter-current conditions, thereby enhancing energy efficiency. The new nominal cooling capacity was 15.4 kW and the new nominal heating capacity was 18 kW. For improved adaption of the heat pump's capacity to the thermal load, two in tandem working compressors were included. [9]

The borehole field consists of $N = 6$ boreholes, aligned in a balanced 2x3 grid, each with a depth of $D = 50$ m, separated at distance $s = 3$ m from each other, and connected in parallel. The diameter of each borehole is given as $d_{bh} = 150$ mm, and each borehole contains one polyethylene U-tube with an internal diameter of $d_{tube} = 25.4$ mm and a distance of $s_{tube} = 70$ mm separating upward and downward pipe. The space between the tubes and the borehole wall is filled with sand and sealed with a layer of bentonite on top of it, to prevent intrusion of pollutants. A laboratory analysis on soil samples was carried out to estimate the thermal properties of the ground, yielding a thermal conductivity $\kappa = 1.43$ W/m and a volumetric heat capacity $C_{soil} = 2.25$ MJ/m^{3}K. However, these values contain an uncertainty of around 20% due to inherent inhomogenities in the soil. In addition, the upper part of the soil is expected to exhibit higher thermal conductivity due to the saturation of the ground with water, given that the phreatic water level is approximately 3.5 m. The sensors installed to enable optimal system operation include: temperature sensors, mass flow meter, and power meter. There are two temperature sensors for each borehole, measuring the inlet and outlet temperature of the water. In addition, in some of the boreholes, there are temperature sensors installed at different depths to measure the ground temperature at the midpoint between two pipes of the U-tube [9].

3.1.2. Theoretical Model of the System

The general underlying Thermodynamics of the heat exchange in the boreholes can be given by the following simplified energy balance.

$$\begin{aligned} \dot{Q}_{\text{heat}}(t) &= \dot{Q}_{\text{ground}}(t) \\ \dot{m}(t) \cdot c_p \cdot |T_{\text{out}}(t) - T_{\text{in}}(t)| &= \frac{1}{R_{\text{total}}} \cdot A \cdot |T_{\text{bw}}(t) - \frac{T_{\text{in}}(t) + T_{\text{out}}(t)}{2}|, \end{aligned} \tag{3.1}$$

where $\dot{Q}_{\text{heat}}$ represents the heat transferred to or from the heat exchanger, A represents the area over which the heat transfer takes place, and $\dot{Q}_{\text{ground}}$ is the heat extracted from the ground. The fluid inlet and outlet temperatures are denoted as T_{in}, respectively T_{out}, the specific heat capacity of the fluid is c_p, and R_{total} represents the total thermal resistance of the system. Under the simplification of a steady-state assumption, at one step of time t, the heat fluxes must equal. Due to the lack of additional parameters in the dataset, a constant thermal resistance R_{total} of the surrounding materials was assumed. This neglects the fact that the thermal resistance of the grout and the ground may vary with both time and depth. Furthermore, an average temperature of the fluid was assumed (perfect mixing), and the effects of external influences, such as the neighboring boreholes were neglected in Equation 3.1.

3.1.3. Dataset Analysis and Preprocessing

The dataset provided by Ruiz et. al [9] consists of three different subsets: system data, borehole data, and reference data. The system data contains parameters of the overall GSHP systems, such as mass flow rates and inlet-/outlet temperatures of both internal and external circuits, and electric power consumption of the heat pump. The borehole data contains measures from all sensors installed within the BHE, that is, the inlet and outlet temperature of each individual borehole, as well as the temperature in one of the six boreholes at different depths. A visualization of the original temperature sensor measurements for the borehole wall temperature of borehole 6 at different depths is given in Figure 3.2.

Apparently, there are many unrealistic measurements in the data, as it can be assumed that the ground at 0 - 50 m depth does not reach temperatures of neither $+90$ nor -20 °C. Furthermore, the reference data contains information about the operational state of the system as well as a Boolean column 'Representative data', indicating whether the values of the corresponding row (DateTime) could contain reliable values.

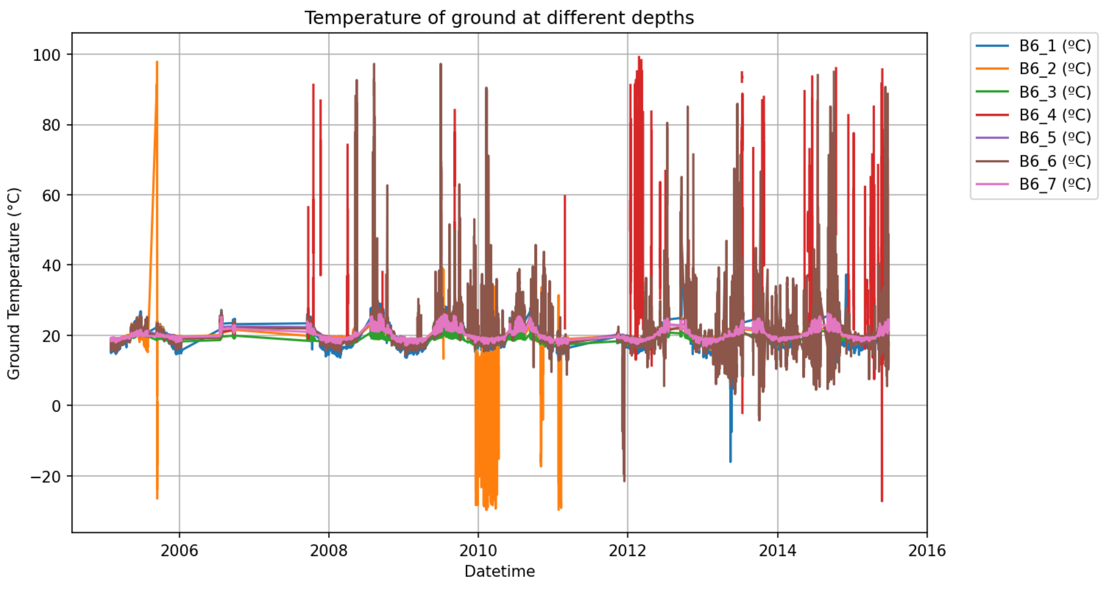


Figure 3.2.: Temperature sensor measurements in borehole 6, at various depths

To achieve results that closely resemble real-world scenarios, we opted to incorporate only features in our model that are typically accessible in practical settings. These features (sensor points) are given in Table 3.1.

Abbreviation	**Unit**	**Explanation**
$T_{\text{in, EC}}$	°C	Temperature of the water at the inlet of the BHE (EC for external circuit of the heat pump)
$T_{\text{out, EC}}$	°C	Temperature of the water at the outlet of the BHE
$\dot{m}_{\text{EC}}$	kg/h	Mass flow rate in the external circuit
DateTime	-	Cyclic encoded to multiple features, as explained in subsection 3.1.3

Table 3.1.: Original feature set

In order to implement a physically motivated loss into the ML algorithms, the borehole temperature are required as an additional feature. This is more than what is typically given in a BHE installation, but it turns out to be highly problematic to add meaningful equations without this parameter. To create the feature T_{bw}, we used that the temperatures at the borehole wall for different depths were available in the dataset for at least one of the six boreholes, and assumed that this temperature would be the same for each of the boreholes in the borehole field. Due to the assumption for a continuous borehole wall temperature, which is valid for a high enough mass flow rate according to Eskilson (as explained in Sec. 3.1.2), we can calculate the average temperature of the borehole wall $T_{\mathrm{bw}}(t)$ at a time step t as follows:

$$T_{\mathrm{bw}}(t) = \frac{1}{L} \sum_{i=1}^{7} l_i T_{\mathrm{bw,\ i}}(t), \tag{3.2}$$

where l_i represents the influence length segment of the corresponding temperature T_i of the sensor located at that point. The temperature sensors are located at equally spaced depths. The first one placed at -2.5 m, corresponds to the start of the active borehole length, and the last one placed at -47.5 m corresponds to the end of the active borehole length. All the features of this extended feature set are listed in Table 3.2.

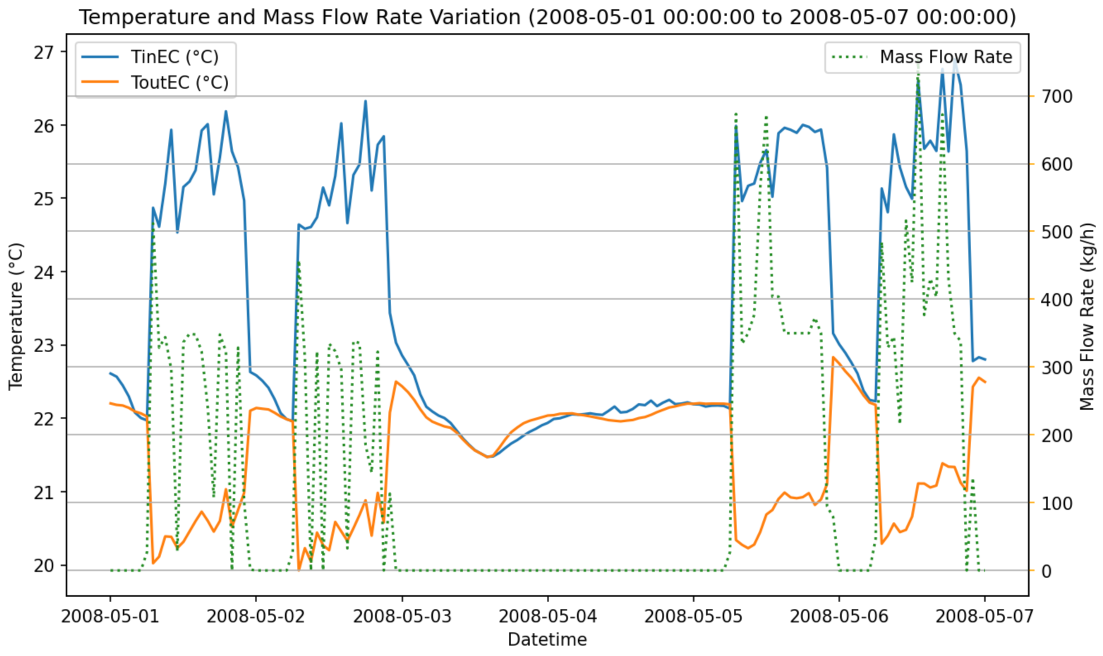
Abbreviation	**Unit**	**Explanation**
$T_{\mathrm{in,\ EC}}$	°C	Temperature of the water at the inlet of the BHE (EC for the external circuit of the heat pump)
$T_{\mathrm{out,\ EC}}$	°C	Temperature of the water at the outlet of the BHE
$\dot{m}_{\mathrm{EC}}$	kg/h	Mass flow rate in the external circuit
T_{bw}	°C	Temperature of the borehole wall
DateTime	-	Cyclic encoded to multiple features

Table 3.2.: Extended feature set for physics-based loss

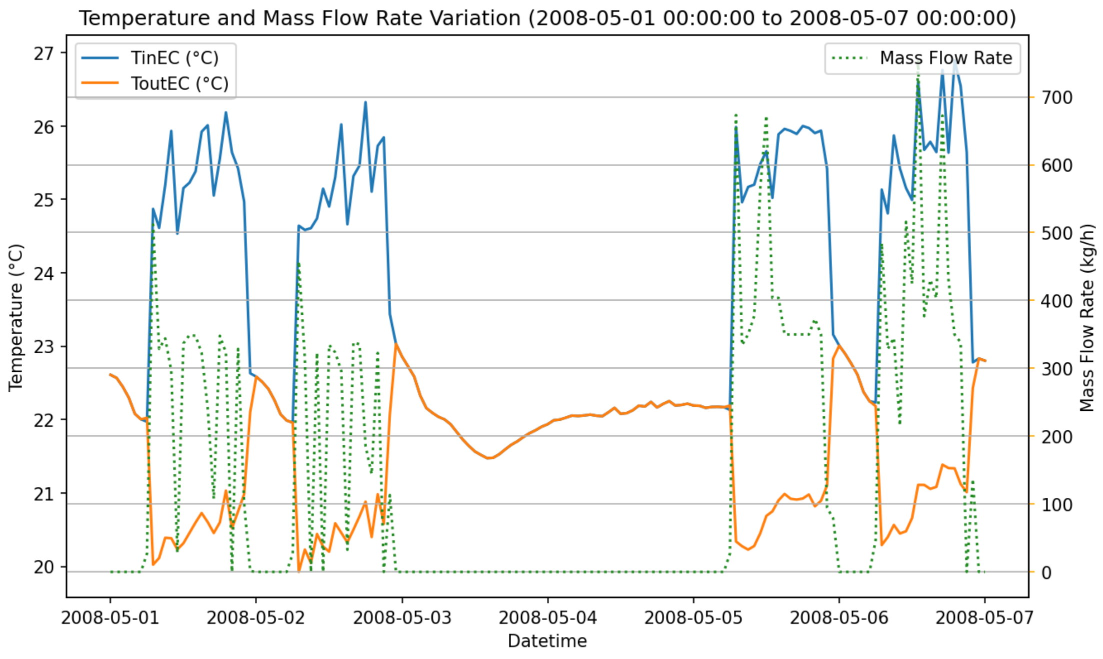
Merging the datasets, resampling over 60-minute intervals, and removing non-representative data yields our preliminary dataset. As a next step, fundamental knowledge about the system is used to preprocess the dataset. First, all temperature values of the fluid (T_{out}, T_{in}) outside the range [-30°C, 100 °C] are set to nan. Then, mass flow rates below 0 are replaced with nan entries, since the flow in the pipes cannot go backward. In addition, for periods when the system is not working longer than 3 hours (e.g. weekends, and night-time), the outlet temperature is set equal to the inlet temperature, because there is no need to predict the temperature for the system if it is not in operational mode. To indicate this inferred operational status of the pump, we add a binary feature called "pump status", which will also be used for the evaluation (compare subsection 3.2.5). The operating hours of the systems are (theoretically) defined from 7 am to 9 pm during 5 days of the week. An exemplary visualization of this slightly artificial modification of output temperatures is represented in Figure 3.3.

Plotting the inlet and outlet temperatures of the whole dataset over time discloses the large number of nan values, which mostly result from the exclusion of non-representative data and physically non-realistic data records. A visualization is given in Fig. 3.4.

As a next step, the DateTime features, originally given in the format 'YYYY-mm-dd HH:MM:SS', need to be cyclically encoded. The idea is to extract as much meaningful information from the feature as possible, that can be effectively interpreted by the ML



(a) Original T_{out}



(b) T_{out} after modification

Figure 3.3.: Results of setting T_{out} equal to T_{in} for non-operating states longer than 3 hours

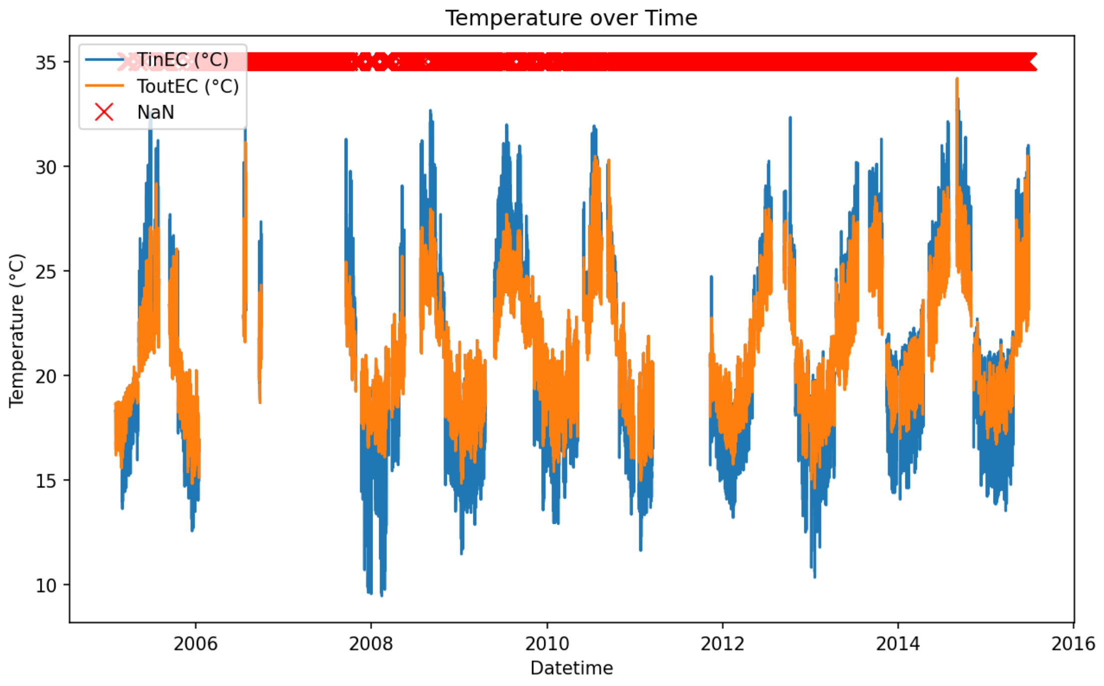


Figure 3.4.: Inlet- and outlet temperatures of the whole time span

models. For example, even though the time step from '2010-12-31 23:59:59' to '2011-01-01 00:00:00' may be very close to each other, which implies that the corresponding values of this time step are likely to be similar, the numbers at this time step change significantly. Therefore, ML models will assume larger changes from one timestamp to the next one in this case. To avoid this, cyclic encoding uses sine and cosine values, effectively representing smooth transitions of consecutive DateTime entries. The encoded DateTime features used for this work are given in Table 3.3.

Table 3.3.: Encoded DateTime features

Original Parameter	**Encoded Parameter**
DateTime	sin(day of year)
DateTime	cos(day of year)
DateTime	cos(hour of day)
DateTime	cos(hour of day)
DateTime	Day of week (0-6)
DateTime	Is Weekday (0,1)
DateTime	Year

Finally, the features are scaled to standardized intervals of $[0, 1]$ in order to avoid features with larger magnitudes predominating the learning behavior of the ML models.

To this end, the scaling (min-max normalization [55]) presented in Equation 3.3 is carried out for each feature in each fold individually. To rule out any information leakages, only the training data is used for fitting the scalers.

$$x_{\text{scaled}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}. \tag{3.3}$$

For the regression models based on a certain look-back width l and a pre-defined prediction horizon of length p, it is necessary to generate windows from the original data frame. Each window has a total width w, given as $w = l + p$. The goal is to train a regression model so as to predict T_{out} for the next p time steps (hours), given the input parameters of the prior l hours and the future inputs T_{in} and $\dot{m}$, defined by the control regime. Since the implemented ML models are not inherently able to deal with nan values in the data, there are two options:

(i) Discard all windows containing nan entries:
This means that for each window of size = total width $w \times k$ features if any of the cell entries contains a nan value, the whole window is discarded from the dataset. Apparently, this leads to significantly downsized datasets, which could be a problem for achieving successful training of the models.

(ii) Impute nan entries as an additional preprocessing step:
We implement an imputation procedure where first, an additional help feature is created to indicate whether more than $x = 48$ consecutive (w.r.t time) entries of the data were nan values. If so, we do not perform an imputation because the time span is just too long to impute meaningful values. If the number of consecutive nans was smaller x, we perform the imputation by training a small neural network with (200, 100, 50) hidden layer sizes on the remaining features with a 90/10 train-test-split, using the *scikit-learn* MLP Regressor package. We add a new binary feature 'Is Imputed' indicating whether a value of the corresponding row (time step) was imputed, and use this feature to calculate corrected MAE scores for the final evaluation (details are given in subsection 3.2.5).

After trying both options in a preliminary regression experiment, it was revealed that imputation leads to a noticeable performance increase. Thus, missing value imputation will be included in the preprocessing pipeline for the remaining experiments. A comparison of the preliminary experiment results with and without imputation is given in Fig. 3.8.

3.2. Implementation

For the implementation of our Machine Learning model in combination with additional Physical information, we opt for an Autoregressive LSTM as our central approach. In addition, we implement two simple baseline models to compare our approach to and get an impression of the complexity of the tackled task. In essence, we will perform two experiments:

(1) Lookback width of 1 week $l = 7$ days $= 168$h and prediction horizon of $p = 7$ days $= 168$h

(2) Lookback width of 1 week $l = 14$ days $= 168$h and prediction horizon of $p = 14$ days $= 168$h

3.2.1. Implementation of the Baseline Methods: Constant and Linear Model

The constant baseline model simply predicts a constant value over the entire prediction horizon. This constant value is given by the last target temperature in the input sequence, i.e.

$$\hat{y}^{\text{cb}}_{t+\tau} = y_{t-1}, \tag{3.4}$$

where the superscript "cb" denotes "constant baseline". The linear baseline model uses the input sequence (look-back data) to predict the entire output sequence (prediction horizon) for the target feature directly in one step. It was created with the TensorFlow [56] library. The model is trained with the Adam optimizer, using a batch size of 16, 20 maximal epochs for training in the hyperparameter tuning process, and 50 maximal epochs for training in the test loop. The hyperparameters tuned by grid search are the learning rate and the patience for early stopping. More details are provided in subsection 3.2.4.

3.2.2. Implementation of a Standard Autoregressive LSTM

The central model used in this thesis for forecasting the borehole output temperatures is an autoregressive LSTM model, that iteratively forecasts one time step ahead. The model architecture comprises an LSTM cell encapsulated within an RNN layer, and a dense layer with a kernel regularizer ($\text{l2} = 0.01$) that transforms the LSTM output into predictions. The chosen architecture allows for a warm-up phase, where the LSTM state is initialized by the input sequence to generate a first prediction, and a prediction phase, where the model concatenates the previous prediction with external features and uses them to produce the next prediction. The prediction phase continues iteratively until the desired number of output predictions (prediction horizon) has been reached. The model is strained using the Adam optimizer, MSE loss, a batch size of 16 and a maximum of 20 epochs for hyperparameter tuning. For the final evaluation on the test set a maximum of 50 epochs is used. The hyperparameters are: learning rate, patience (for the early stopping mechanism), and the number of LSTM units (compare subsection 3.2.4).

3.2.3. Implementation of the Custom Loss

To incorporate additional information we have about the given data, such as domain knowledge and fundamental thermodynamics principles, the traditional loss function is extended by both a constraint-loss and a physics-loss term:

$$\mathcal{L} = w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{phys}}\mathcal{L}_{\text{phys}} + w_{\text{const}}\mathcal{L}_{\text{constraints}}, \tag{3.5}$$

where the weights $w_{\text{data}}, w_{\text{phys}}, w_{\text{const}}$ are hyperparameters specifying the influence of the individual loss terms.

Data Loss

The data loss is calculated as the mean squared error between the model outputs for T_{out}^{model} and the true (experimental) values from the data T_{out}^{true}.

$$\mathcal{L}_{\text{data}} = \frac{1}{N}\sum_{i=1}^{N}(T_{\text{out,i}}^{\text{model}} - T_{\text{out,i}}^{\text{true}})^2 \tag{3.6}$$

Constraint Loss

The constraints are given in the form of statistical knowledge about the system. From the given training data, it can be inferred that the variation of the temperature T_{out} from one time step to the next typically lies within a certain range. In other words, there are usually no big jumps in consecutive values. To make use of this information, the mean μ and standard deviation σ of these consecutive temperature differences can be calculated as:

$$\begin{aligned} \mu(\Delta T) &= \frac{1}{N}\sum_{i}^{N}(\Delta T_{\text{out,i}}) = \frac{1}{N}\sum_{i}^{N}(T_{\text{out,i}} - T_{\text{out,i-1}}) \\ \sigma(\Delta T) &= \sqrt{\frac{1}{N}\sum_{i}^{N}(\Delta T_{\text{out,i}} - \mu(\Delta T_{\text{out}}))^2}. \end{aligned} \tag{3.7}$$

To only penalize rather strong deviations from the usual behavior, we opted for an acceptance range defined as:

$$I_{\text{acc}}^{\Delta T} = [0, \mu(\Delta T_{\text{out}}) + 2\sigma(\Delta T_{\text{out}})]. \tag{3.8}$$

In addition, we make use of physical domain knowledge. In particular, we know that the output fluid (water) of the pipes will always have a temperature T_{out} between 10 and 40 °C. This can be formulated with an acceptance interval

$$I_{\text{acc}}^{T_{\text{out}}} = [T_{\text{out,min}}, T_{\text{out,max}}] = [10, 40], \tag{3.9}$$

leading to a total constraint loss of the form

$$\begin{gathered} \mathcal{L}_{\text{constraints}} = \frac{1}{M}\sum_{j}^{M} a \cdot |\Delta T_{\text{out,j}}| + \frac{1}{P}\sum_{p}^{P} b \cdot |T_{\text{out,p}} - T_{\text{out,min}}| + \frac{1}{Q}\sum_{q}^{Q} b \cdot |T_{\text{out,q}} - T_{\text{out,max}}|, \\ \text{for } \Delta T_{\text{out,j}} \notin I_{\text{acc}},\ \text{for } T_{\text{out,p}} < T_{\text{out,min}},\ \text{for } T_{\text{out,q}} > T_{\text{out,max}} \end{gathered} \tag{3.10}$$

with a and b as tuneable hyperparameters.

Physics Loss

Regarding the physics loss, we are strongly restricted in our choice of equations due to the small number of parameters and sensor points given. Based only on the mass flow rate $\dot{m}(t)$, the temperature input $T_{\text{in}}(t)$, the constructed average borehole wall temperature $T_{\text{bw}}(t)$, and the encoded date- and time-features apart from some constant parameters of the system, it is impossible to implement a PDE describing the heat transfer over time and/or space. Therefore, the only possibility to incorporate Thermodynamics was using the simplified steady-state energy balance presented in Equation 3.1. Using this equation, the physics loss can be formulated as

$$\mathcal{L}_{\text{phys}} = \frac{1}{K}\sum_{k=1}^{K}\left\{\dot{Q}_{\text{heat,k}}(t) - \dot{Q}_{\text{ground,k}}(t)\right\} = \frac{1}{K}\sum_{k=1}^{K}\left\{\dot{m}\cdot c_p \cdot |T^{model}_{\text{out,k}}(t) - T_{\text{in,k}}(t)| - \frac{1}{R_{\text{total}}}\cdot A \cdot \left|T_{\text{bw,k}}(t) - \frac{T_{\text{in,k}}(t) + T^{model}_{\text{out,k}}(t)}{2}\right|\right\}, \tag{3.11}$$

where $k \in K$ are the samples (time steps) where the heat exchanger is operating, indicated by $\dot{m}(t) > 1$ with 1 as a threshold to account for the noise of the mass flow meter sensor. To ensure that the physical loss from the energy balance yields meaningful results for the given dataset, a pre-check was performed where the loss was calculated for the true target values $T_{\text{out}}(t)$ of the dataset. To increase model performance by improving generalization and avoiding over-fitting, the incorporated Physical relations should be able to (at least somewhat) resemble the measured temperature output of the BHE. To this end, the heat fluxes $\dot{Q}_{\text{heat}}$ and $\dot{Q}_{\text{ground}}$ calculated from the experimental data are plotted over time. Under ideal conditions, these two values should match. The results of this experiment are presented below in Figure 3.5. It was found that the heat fluxes differed significantly, indicating that the equation does not capture the behavior of the boreholes at all. After careful inspection of the calculation, no error could be detected. As a next step, we wanted to rule out that the parameter $k = \frac{1}{R_{\text{total}}}$, obtained from the dataset, was incorrect. Therefore, we calculated k from Eq. 3.1 with the experimental data and plotted it over the given time span (compare Figure 3.6). As can be observed from the graph, the value of k fluctuates strongly over time. This indicates that the energy balance equation cannot be fulfilled for any constant values of R_{total} and A. Since all other parameters in the equation were directly taken from the dataset at hand, it needs to be concluded that the steady-state energy balance, with the numerous simplifications and assumptions we had to make due to the limited availability of parameters/sensor values, is clearly not able to model the thermal behavior of the boreholes with sufficient precision. Hence, using the aspired physics loss cannot yield an increase in the ML model performance and will not be used in subsequent experiments.

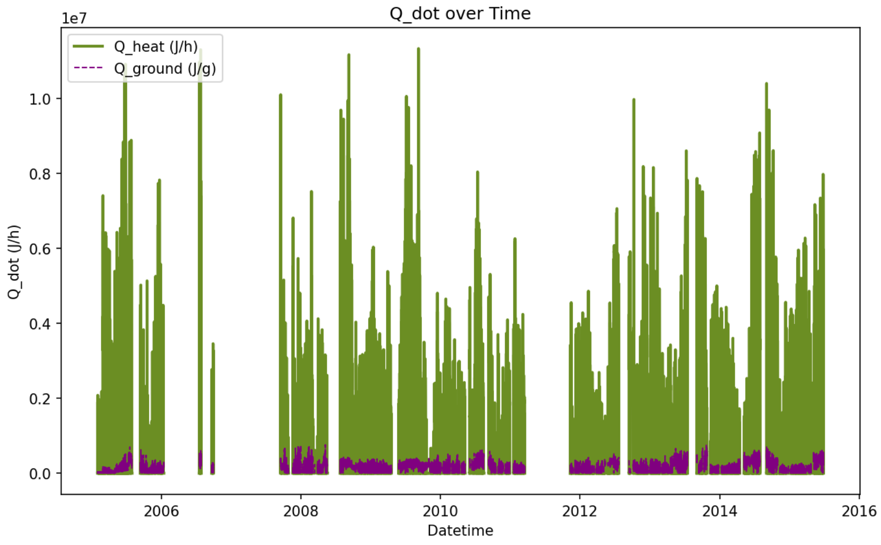


Figure 3.5.: Plot of calculated $\dot{Q}_{\text{heat}}$

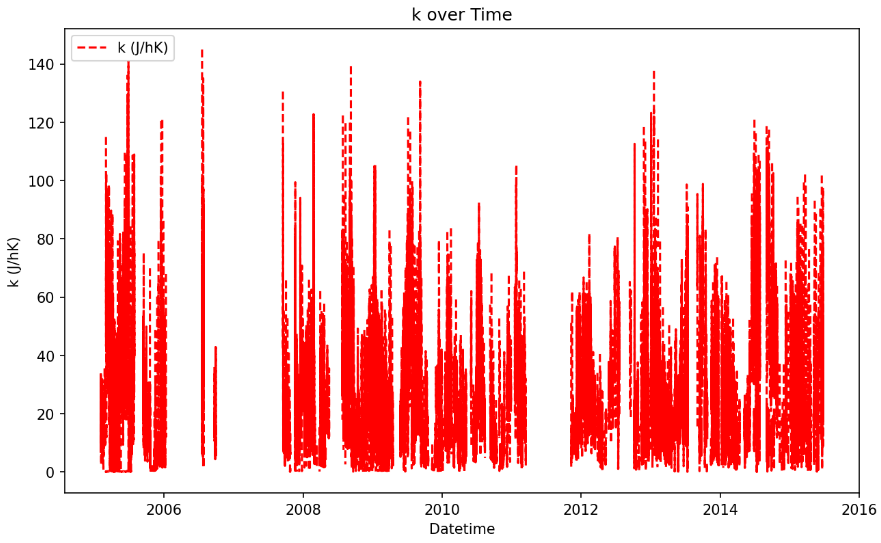


Figure 3.6.: Plot of calculated $k = \frac{1}{R_{\text{total}}} \cdot A$

3.2.4. Hyperparameter Tuning

A suitable choice of the hyperparameters for our model is the key to achieving satisfactory performance. Therefore, we decided to carry out hyperparameter tuning to make the most of the model's learning capabilities. Since the performance is also reliant on sufficient training data, we choose to perform the hyperparameter tuning on data sets as large as possible, while still ensuring that there is no information leakage with time-series cross-validation. In particular, for each fold of the cross-validation, we split the fold into training-, validation, and testing sets and used the training- and validation sets for the hyperparameter tuning process. To not completely overburden the necessary computational resources, we opt for a grid search on a limited hyperparameter space, as given in Table 3.4. For the linear baseline model, only the hyperparameters 'learning rate' and 'patience' are relevant, while for the constant baseline model, the hyperparameters have no influence at all since there is essentially no learning taking place. The test set is held out during this process and only used for the final evaluation with the best parameters found from the hyperparameter tuning.

Hyperparameter	**Parameter Space**
learning rate	0.0005, 0.0002
patience	4, 8
w_{data}	0.5, 0.8, 1.0
w_{const}	0.5, 0.2, 0.0
a	50, 300
b	0.1, 0.5
LSTM units	100, 150

Table 3.4.: Hyperparameter space for hyperparameter tuning. For parameter definitions refer to section 3.2

3.2.5. Evaluation Method

To avoid information leakage and increase the reliability of the results, we perform a 5-fold time series cross-validation. In addition, the data is split into train-, validation- and test-set with split sizes according to Table 3.5 and Table 3.6. The slightly varying sizes of the validation set over the different folds arise due to the fact that we dropped overlapping windows (and the overlaps vary since many windows were discarded even before the splitting due to missing values). First, the hyperparameter tuning is performed, where we train the models on the training data and evaluate them on the validation data. Then, the hyperparameters for which the individual model performed best (averaged across all folds) are selected for the final evaluation cycle, where the models are trained on both training data and validation data, and evaluated on the until-that-point hold-out test set.
We opt for MAE as an overall evaluation metric, as it provides a clear and reliable measure of errors, ensuring consistency across all forecasting windows, thereby offering a straightforward interpretation that aligns well with the nature of our data. To account for values where we set $T_{\text{in}} = T_{\text{out}}$ due to the pump not being in operational mode, we also calculate corrected MAE-Scores, where we assign zero weight to the predictions where the pump status is off. In addition, also for the originally missing value that we imputed, we set the weights to zero for the corrected scores.
For further evaluation, we additionally compare the results of our approach to the purely theoretical model from Ruiz et al. [10] introduced in Section 3.3.

Fold	**Train set**	**Validation set**	**Test set**	**Total set**
1	12185	776	1111	14072
2	13296	776	1111	15183
3	14205	978	1111	16294
4	15202	1092	1111	17405
5	16626	779	1111	18516

Table 3.5.: Split sizes for experiment 1, i.e. number of windows (each of total width = look-back window + prediction horizon = 2 weeks) for the subsets in the different folds

Fold	**Train set**	**Validation set**	**Test set**	**Total set**
1	6299	195	866	7360
2	6808	552	866	8226
3	7672	554	866	9092
4	8226	866	866	9958
5	9717	241	866	10824

Table 3.6.: Split sizes for experiment 2, i.e. number of windows (each of total width = look-back window + prediction horizon = 4 weeks) for the subsets in the different folds

3.3. Theoretical Reference Model

The authors of our investigated dataset developed an extensive, purely theoretical model[10] to describe the behavior of the BHE at the site in Valencia. The results of this physical model will be used as a reference to compare our proposed Physics-informed LSTM to. According to Ruiz-Calvo et. al [10], the theoretical model of the behavior of the BHE system can be decoupled into long-term and short-term responses. For the long-term part, Eskilson 1987 [41] proposed a so-called g-function model, which has undergone numerous validation and is widely applied. For the short-term response, a thermal network approach is used, which has already been proven[38] to achieve good simulation results for the investigated BHE dataset from Universitat Politècnica de València.

3.3.1. Long-term: g-function Model

The g-function model is a numerical method for BHE response calculations, based on a finite differences algorithm, and relates the change in the borehole wall temperature $T_{\text{bw}}(t)$ over time to the initial undisturbed ground temperature T_g, given a constant heat extraction/injection rate (heat flow) $\dot{q}$. The function depends on the geometrical properties of the borehole field, expressed as ratios: $\frac{r_b}{H}$, $\frac{B}{H}$, $\frac{I}{H}$, where r_b is the borehole radius, B the spacing between two neighboring boreholes, I is the inactive upper part of the boreholes and H represents the active borehole length. In addition, the g-function is a function of a non-dimensional ratio t/t_s, with t_s being the characteristic time $t_s = \frac{H^2}{9\alpha}$, and α representing the thermal diffusivity. Then, the relation is given as

$$T_{\mathrm{bw}}(t) - T_g = \frac{\dot{q}}{2\pi k} \cdot g\Big(\frac{t}{t_s}, \frac{r_b}{H}, \frac{B}{H}, \frac{I}{H}\Big), \tag{3.12}$$

with k specifying the thermal conductivity of the ground [10]. The boundary conditions for Eq. 3.12 can either be formulated for (i) a constant heat load per unit length segment of the borehole, or for (ii) a uniform temperature along the borehole wall. As also investigated by Eskilson[57], for boreholes connected in parallel with sufficiently large mass flow rate, the boundary condition (ii) will be approached. The formulation of the g-function requires inputs in the form of constant load steps $\dot{q}_i$, thus, a discretization of the continuous real-world thermal load into constant load blocks is necessary. One thermal load block $\dot{q}_n$ is calculated as the sum over all preceding load steps (differences between two consecutive thermal load blocks $\dot{q}_i - \dot{q}_{i-1}$), thus

$$\dot{q}_n = \sum_{i=1}^{n} (\dot{q}_i - \dot{q}_{i-1}) \tag{3.13}$$

According to [10], the g-function evaluated at a specific time-step can then be rewritten as:

$$g\Big(\frac{t}{t_s}, \frac{r_b}{H}, \frac{B}{H}, \frac{I}{H}\Big)|_{t=t_i} = g\Big(ln(\frac{t-t_i}{t_s})\Big) \tag{3.14}$$

and the borehole wall temperature, representing the cumulative responses to the individual load steps until the current time step is calculated as:

$$T_{bw}(t) = T_g + \sum_{i=1}^{n} \frac{\dot{q}_i - \dot{q}_{i-1}}{2\pi k} \cdot g\Big(ln(\frac{t-t_i}{t_s})\Big) \tag{3.15}$$

For the coupled model, the authors used the output of the g-function model to calculate the reset temperature for the ground and grout nodes, which is then implemented as the initial start temperature for the B2G model for each day. The reset time corresponds to the operation start time of the system on a working day.

3.3.2. Short-term: B2G Model

The short-term model, introduced as "Borehole-to-Ground (B2G) Model"[58, 59] is exclusively concerned with the local heat transfer, taking place between the borehole, the fluid, and the part of the ground directly surrounding the borehole. The B2G model builds upon the so-called "thermal network approach", which describes the heat exchange by representing the borehole and surrounding ground as a series of temperature nodes, connected by thermal resistances. The higher the required accuracy of the thermal network model, the more temperature nodes are required, thereby vastly increasing the complexity of the approach. To reduce the computational costs, but retain a high accuracy, the B2G model was introduced to reproduce the short-term response of a single U-tube BHE for the water temperature throughout the pipe. After vertically discretizing the borehole, thermal networks are introduced to account for the radial heat transfer at each temperature node. At each node depth, a total of five thermal capacitances and six thermal resistances are considered for the investigated system[38]. In particular, heat conduction in vertical direction is neglected, while the thermal properties of pipes, ground, and grout are considered. An illustration of the thermal network is presented below in Fig. 3.7, where two distinct capacitance nodes are given for the grout depending on the location of the pipes, and the U-tube is modeled by two distinct nodes for upward and downward fluid flow.

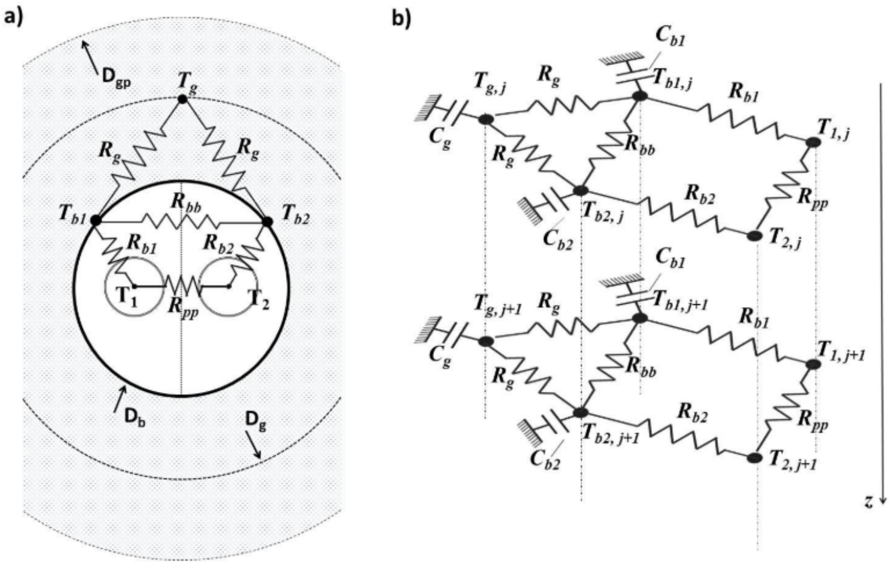


Figure 3.7.: (a) 2D model and (b) 3D model of the thermal network model for the analyzed dataset[38]

The energy balance for all nodes can then be written (in transient state) as[10, 38]:

$$\begin{aligned}
\frac{\partial T_1(z)}{\partial t} &= -v\frac{\partial T_1(z)}{\partial z} - \frac{1}{C_f}\Big(\frac{T_1(z) - T_{b1}(2)}{R_{b1}} + \frac{T_1(z) - T_2(z)}{R_{pp}}\Big)\\
\frac{\partial T_2(z)}{\partial t} &= -v\frac{\partial T_2(z)}{\partial z} - \frac{1}{C_f}\Big(\frac{T_2(z) - T_{b2}(2)}{R_{b2}} + \frac{T_1(z) - T_2(z)}{R_{pp}}\Big)\\
C_{b1}\frac{\partial T_{b1}(z)}{\partial t} &= \frac{T_1(z) - T_{b1}(z)}{R_{b1}} + \frac{T_{b1}(z) - T_{b2}(z)}{R_{bb}} - \frac{T_{b1}(z) - T_g(z)}{R_g}\\
C_{b2}\frac{\partial T_{b2}(z)}{\partial t} &= \frac{T_2(z) - T_{b2}(z)}{R_{b2}} + \frac{T_{b1}(z) - T_{b2}(z)}{R_{bb}} - \frac{T_{b2}(z) - T_g(z)}{R_g}\\
C_g\frac{\partial T_g(z)}{\partial t} &= \frac{T_{b1}(z) - T_g(z)}{R_g} + \frac{T_{b2}(z) - T_g(z)}{R_g},
\end{aligned} \tag{3.16}$$

where the parameters are calculated using the boreholes' geometry and the thermophysical properties of the ground, grout, fluid, and pipes (for adopted values, refer to [10]). In Equation 3.16, $T_1(z)$ and $T_2(z)$, respectively, represent the temperature of the fluid in the upward (1) and downward (2) pipe, T_g is the temperature of the ground, C_g is the thermal capacitance of the ground for a given penetration depth D_{pg}, C_f represents the capacitance of the fluid and $C_{b1} = C_{b2}$ the capacitance of the borehole node. The temperatures at the borehole node position are given by T_{b1} respectively T_{b2}, and the thermal resistances are: R_{pp} for pipe-to-pipe, R_{bb} for borehole-to-borehole, R_g for the ground and R_{b1} respectively R_{b2} for the boreholes.

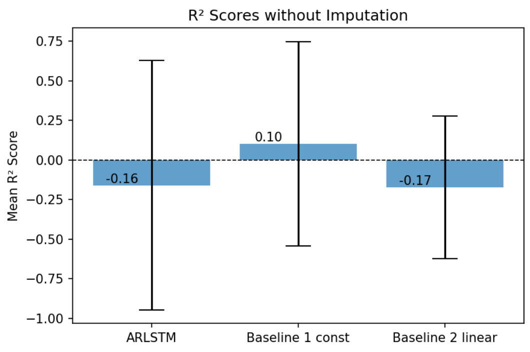
3.4. Results

Missing Value Imputation

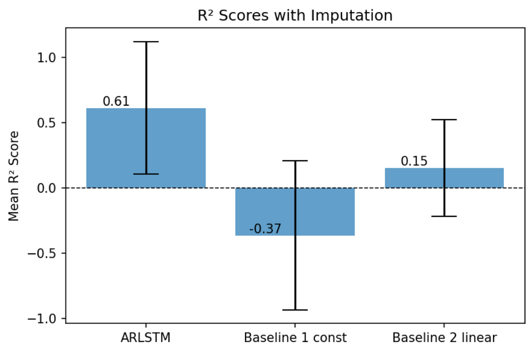
As a first check in order to estimate whether imputation of missing values is useful for the given dataset, we ran the experiment with the standard settings given in Table 3.7 once with and once without imputation. The results of this evaluation are illustrated below in Figures 3.8a and 3.8b.

Table 3.7.: Standard settings for assessment of missing value Imputation

Parameter	Value
Look-back width	7 days = 168 h
Prediction horizon	7 days = 168 h
Max epochs	50
Learning rate	0.0002
Early stopping criterion	Validation loss
Patience	4
Min. delta	0.0001
Start date	2005-02-01 00:00:00
End date	2015-16-26 23:00:00
Custom loss	False
Cross-validation folds	5
Min. Training size	0.7·total dataset size



(a) Results without imputation



(b) Results with imputation

Figure 3.8.: Comparison of prediction results (mean R2 scores and standard deviations represented as error-bars) with and without missing value imputation for the standard ARLSTM (without custom loss) and the baseline models

It can be observed that performing the imputation as an additional preprocessing step increases the performance of the ARLSTM. Therefore, the subsequent experiments were all carried out with missing value imputation enabled.

Table 3.8.: R2 scores for missing value imputation

Target Parameter	**Input Parameters**	**R2 score**
Pump status (on/off)	DateTime-features	0.39
Inlet temperature T_{in} (°C)	DateTime-features, pump-status	0.72
Mass flow rate $\dot{m}$ (kg/h)	DateTime-features, pump-status, T_{in},	0.68
Outlet temperature T_{out} (°C)	DateTime-features, pump-status, T_{in}, $\dot{m}$	0.86

The imputation increased the total number of available windows n (i.e. windows without any nan values) from

$$n_{\text{without imputation}} = 6467$$

to

$$n_{\text{with imputation}} = 18516,$$

which explains the significant improvement of the performance of the regression models.

Hyperparameter Tuning

Experiment (1) with look-back $l = 1$ week and prediction horizon $p = 1$ week is performed with the following fixed parameters in Table 3.9, in addition to the variable hyperparameters shown in Table 3.4. Similarly, for experiment (2) we use the fixed parameters represented in Table 3.10, in addition to the grid search hyperparameters listed in Table 3.4.

Table 3.9.: Fixed parameters for hyperparameter tuning of experiment (1)

Parameter	**Value**
Look-back width	7 days = 168 h
Prediction horizon	7 days = 168 h
Max epochs	20
Early stopping criterion	Validation loss
Min. delta	0.0001
Start date	2005-02-01 00:00:00
End date	2015-16-26 23:00:00
Cross-validation folds	5
Min. Training size	0.7·total dataset size

The best hyperparameters found for each model are then used for the final evaluation on the test set. The corresponding values of the hyperparameters can thus be found in Table 3.11 for experiment (1) and in Table 3.12 for experiment (2).

Table 3.10.: Fixed parameters for hyperparameter tuning of experiment (2)

Parameter	Value
Look-back width	14 days = 336 h
Prediction horizon	7 days = 336 h
Max epochs	20
Early stopping criterion	Validation loss
Min. delta	0.0001
Start date	2005-02-01 00:00:00
End date	2015-16-26 23:00:00
Cross-validation folds	5
Min. Training size	0.6·total dataset size

Test Run for Final Evaluation

The physics loss is not implemented since we assumed a very poor performance due to the strong deviation between Q_{heat} and Q_{ground} (compare Figure 3.5). The parameters used for the final evaluation loop are given below in Tables 3.11 and 3.12.

Table 3.11.: Parameters used for the final evaluation of experiment (1)

Parameter	Value
Look-back width	7 days = 168 h
Prediction horizon	7 days = 168 h
Max epochs	50
Early stopping criterion	Validation loss
Start date	2005-02-01 00:00:00
End date	2015-16-26 23:00:00
Cross-validation folds	5
Min. Training size	0.7·total dataset
Learning Rate	0.0005 (same for lin. Baseline)
Patience	8 (4 for lin. Baseline)
w_{data}	0.8
w_{const}	0.2
a	300
b	0.5
LSTM units	150

In addition, we also intended to determine how the implementation of the constraint loss is affecting the overall performance. Therefore, we carry out the final evaluation run for the ARLSTM also with $w_{\text{data}} = 1.0$ and accordingly $w_{\text{const}} = 0.0$. The best hyperparameters found for only data loss are a learning rate of $5 \cdot 10^{-4}$ and a patience

Table 3.12.: Parameters used for the final evaluation of experiment (2)

Parameter	**Value**
Look-back width	14 days = 336 h
Prediction horizon	7 days = 336 h
Max epochs	50
Early stopping criterion	Validation loss
Start date	2005-02-01 00:00:00
End date	2015-16-26 23:00:00
Cross-validation folds	5
Min. Training size	0.6·total dataset
Learning Rate	0.0002 (0.0005 for lin. Baseline)
Patience	8 (same for lin. Baseline)
w_{data}	0.8
w_{const}	0.2
a	300
b	0.5
LSTM units	100

set to 4 for experiment (1) and a learning rate of $5 \cdot 10^{-4}$ with a patience set to 8 for experiment (2). The results of the final evaluation on the test set, using the best hyperparameters found for each model, are summarized below in Figures 3.9 and 3.10.

To gain further insights into the prediction power of our model, we randomly pick some windows from the test set and plot the model predictions over time. Some representative examples are shown below in Figures 3.11 and 3.12.

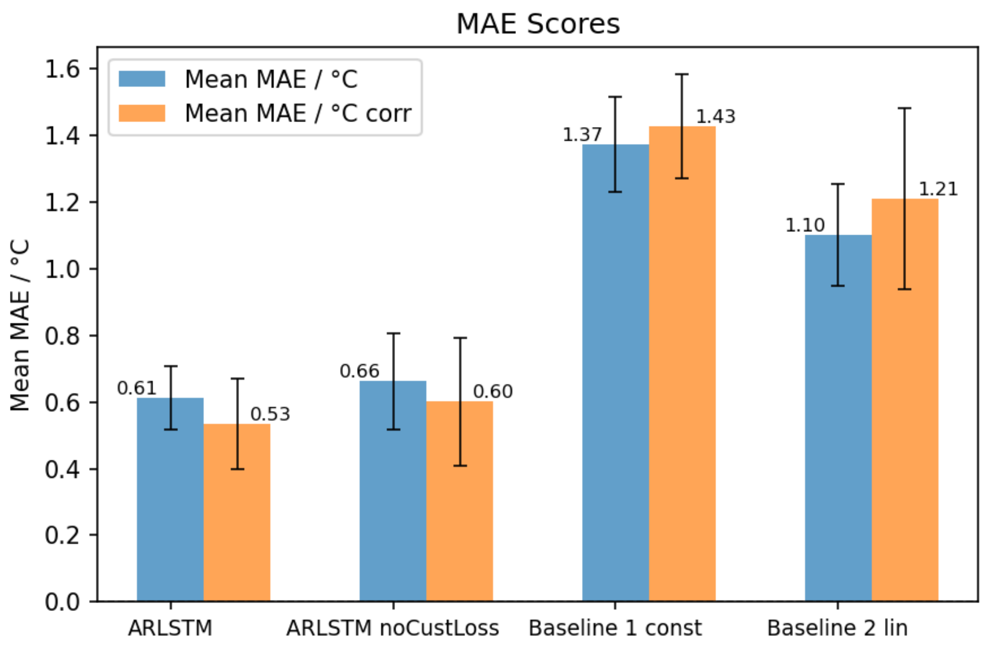


Figure 3.9.: Mean MAE scores over the 5 folds for experiment (1), with standard deviations represented as error bars

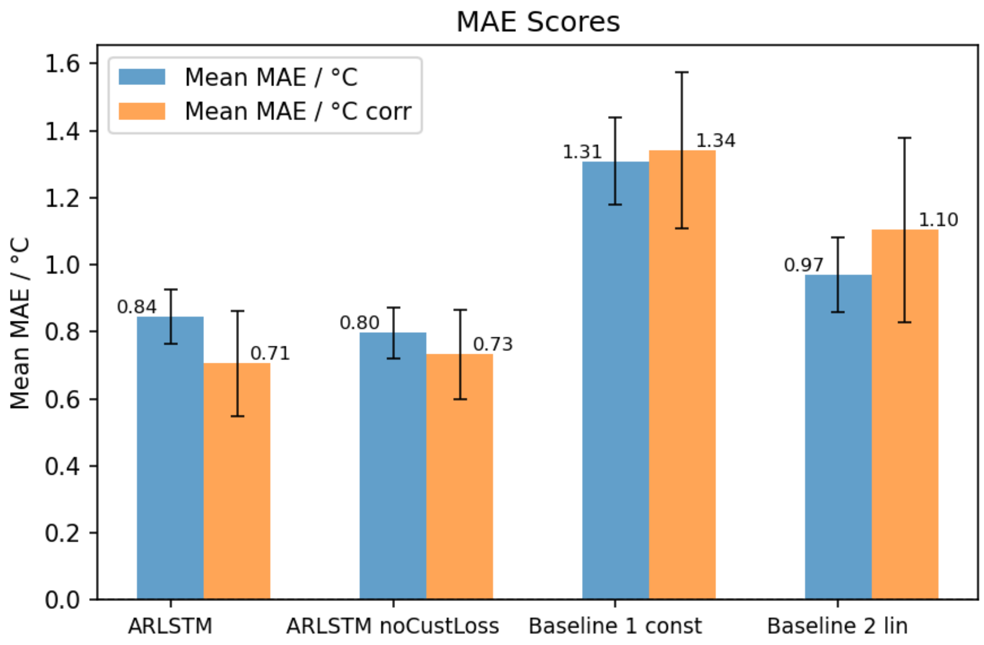
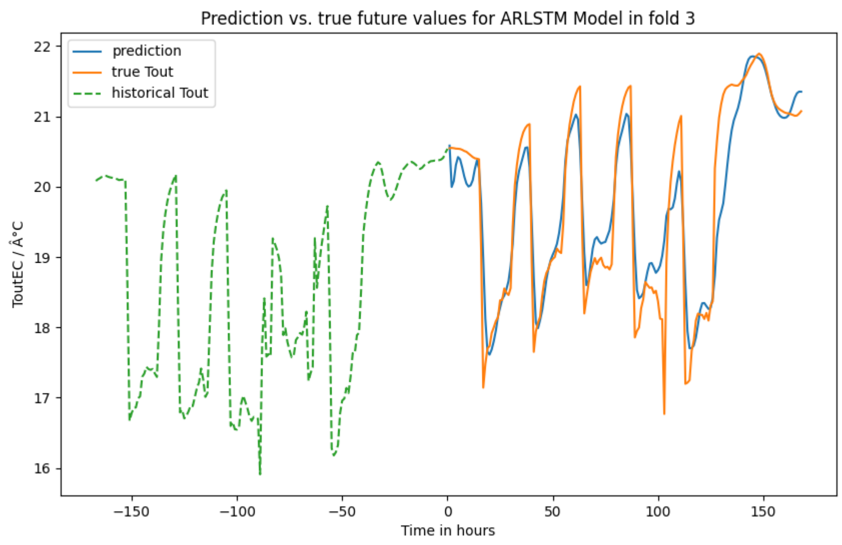
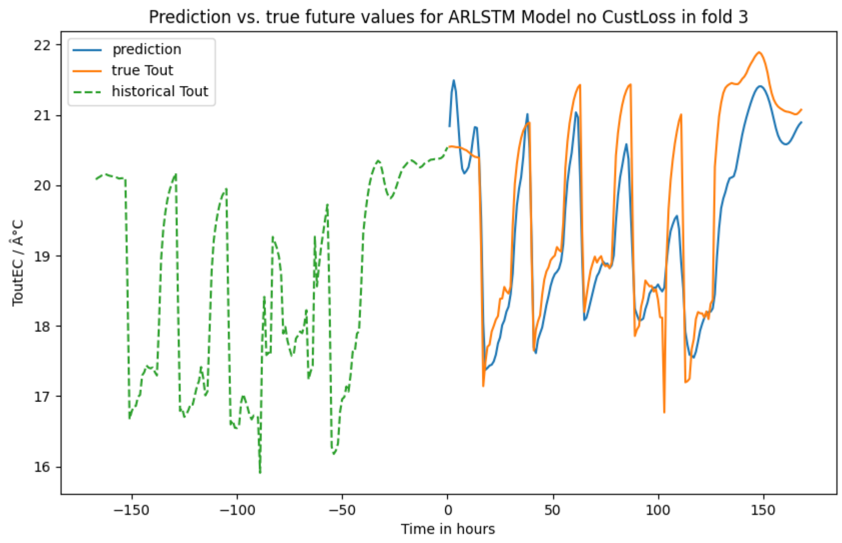


Figure 3.10.: Mean MAE scores over the 5 folds for experiment (2), with standard deviations represented as error bars

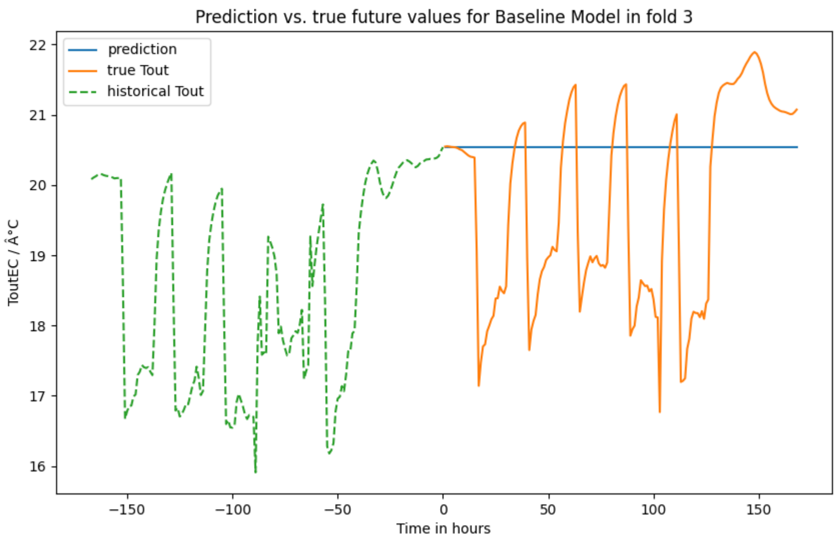


(a) Forecast of ARLSTM with custom loss

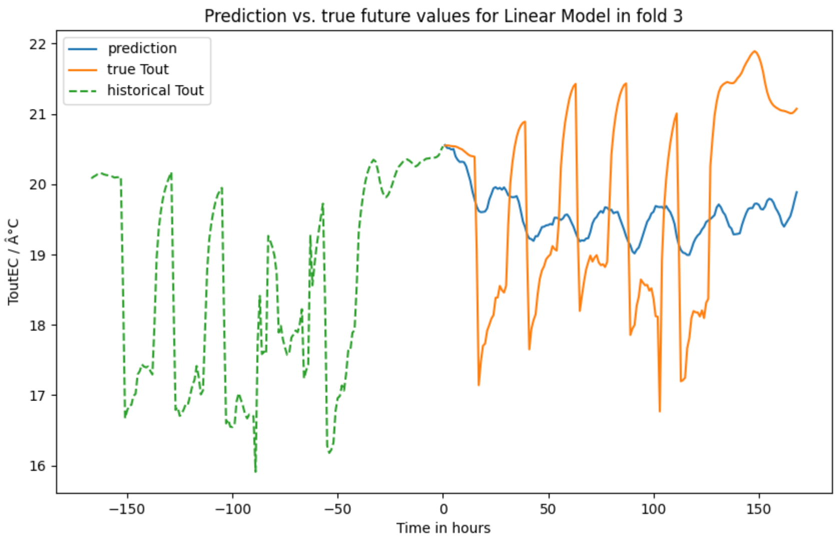


(b) Forecast of ARLSTM model without custom loss

Figure 3.11.: Exemplary prediction plots for experiment (1) - Part 1

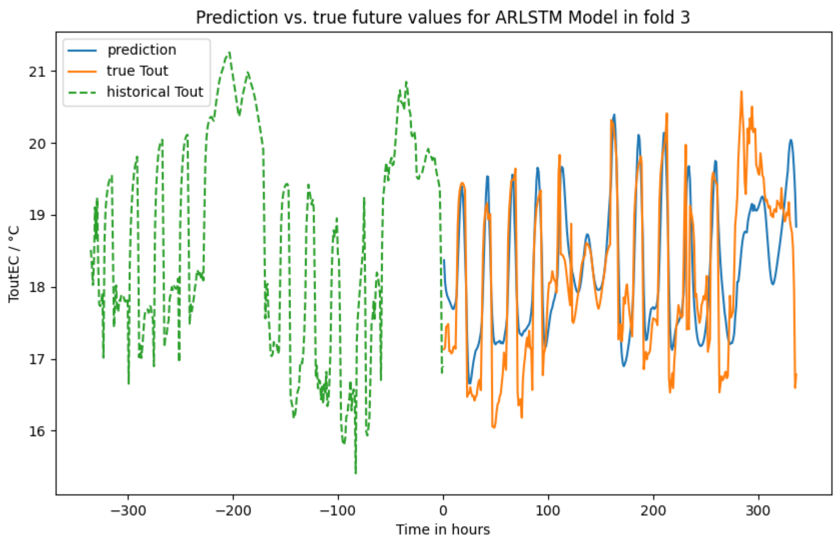


(c) Forecast of constant baseline model

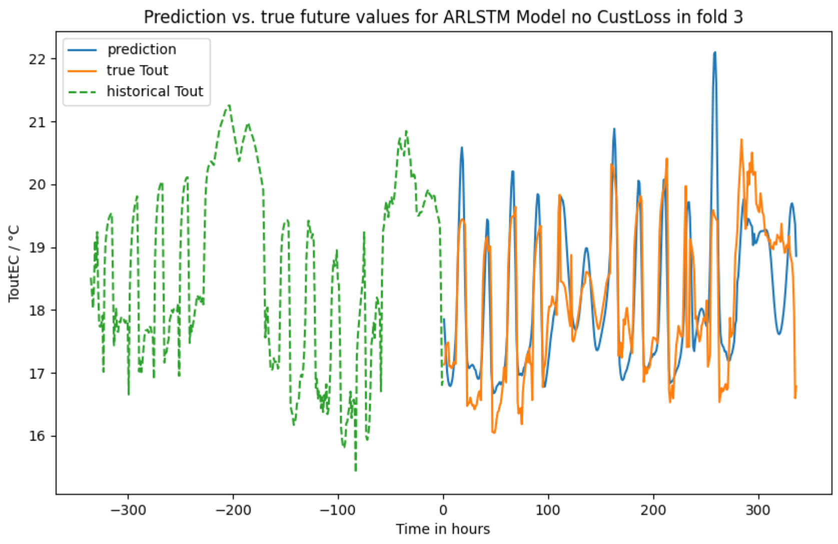


(d) Forecast of linear baseline model

Figure 3.11.: Exemplary prediction plots for experiment (1) - Part 2

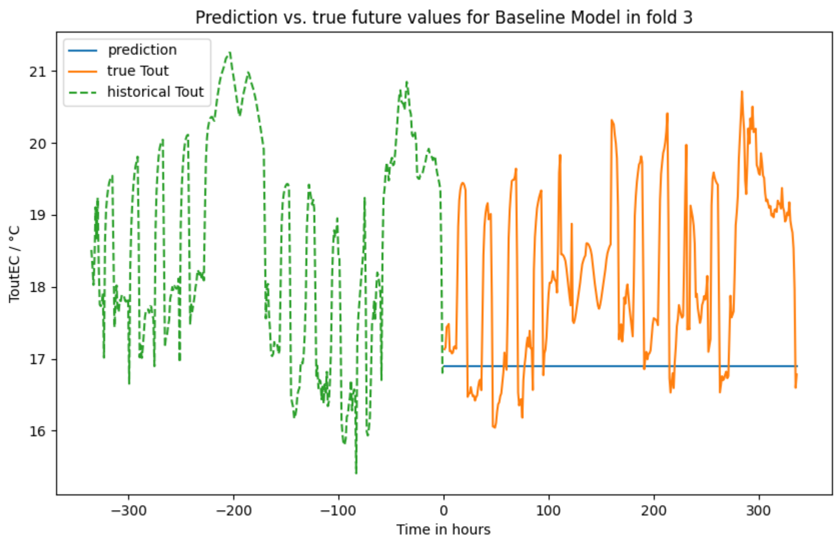


(a) Forecast of ARLSTM with custom loss

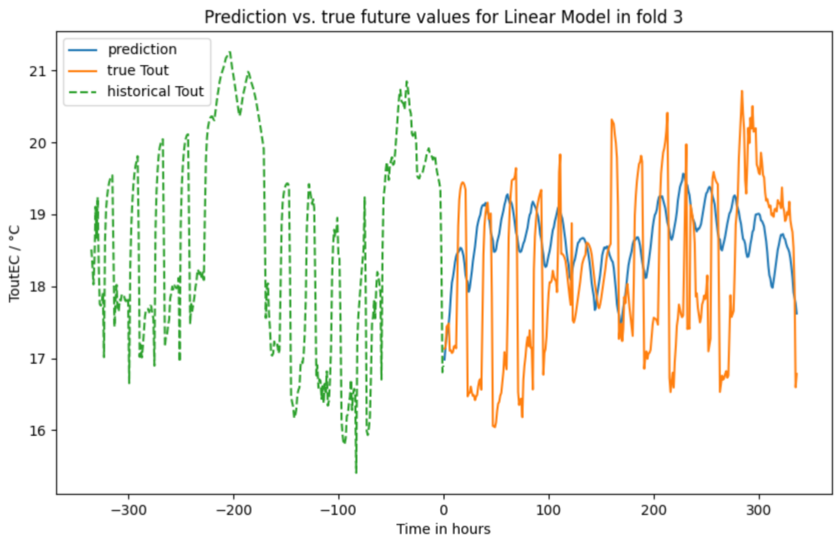


(b) Forecast of ARLSTM model without custom loss

Figure 3.12.: Exemplary prediction plots for experiment (2) - Part 1



(c) Forecast of constant baseline model



(d) Forecast of linear baseline model

Figure 3.12.: Exemplary prediction plots for experiment (2) - Part 2

3.5. Discussion

The results of the pre-experiment of forecasting with and without missing value imputation revealed that imputation drastically increases model performance. This can be attributed to the fact that due to the imputation significantly more windows could be used for the training of the models. In particular, the total number of windows that were used increased from 6467 to 18516. Thus, even though the R2 scores of the imputed values itself, given in Table 3.8, do not ideally resemble the true (missing) data, the performance greatly benefits from the increased number of available training windows.

We emphasize once again that the loss-term $\mathcal{L}_{\text{phys}}$ containing physical equations was not implemented in the code, used model evaluation, as it was anticipated that it would only deteriorate the performance. Our initial expectation was that $\dot{Q}_{\text{heat}}(t) \approx \dot{Q}_{\text{ground}}(t)$, ensuring that the loss term would vanish for correct predictions of $T_{\text{out}}(t)$ from our model (compare Equation 3.11). However, due to the strong deviation between Q_{heat} and Q_{ground} at essentially every time step, as shown in Figure 3.5, we must infer that the simple steady-state energy balance equation does not capture the true behavior of the borehole. Hence, in the final script, only physical constraints were incorporated in the loss function.

The main evaluation metric we considered is the corrected MAE, which was calculated by rescaling the prediction results back to the original ranges and setting the weights of imputed time steps to zero. From the results of the 1-week forecast with the ARLSTM, represented in Figure 3.9, it is evident that the corrected MAE for the ARLSTM with physics-informed loss is 0.7°C lower than the MAE for the ARLSTM without custom loss. The corrected MAE for the constant baseline is 1.43 °C and for the linear baseline 1.21 °C, which effectively illustrates that the errors progressively decrease with increasing model complexity. Furthermore, the prediction plots in Figure 3.11 exemplarily illustrate the goodness-of-fit of the different models. Comparing plots (a) and (b), it can be observed that the ARLSTM without custom loss tends to overshoot, while the predictions of the ARLSTM with the physics-informed loss more closely align with the true target values. This may be due to the fact that the custom loss contains a penalty term for large ΔT_{out} values for consecutive time-steps. In addition, from plot (d) it can be inferred that the linear model is too simple to capture the complex patterns of the borehole output temperature.

The corrected MAE values for the ARLSTM in experiment (2), as represented in Figure 3.10, are somewhat higher than for experiment (1). This aligns well with our expectations since the autoregressive design of the model results in predictions hat deviate further from the true values, the more time steps we move away from the last true $T_{\text{out}}(t = 0)$ value that the model received as input. The corrected MAE for the ARLSTM with custom loss is slightly better (0.71 °C) than the corrected MAE for the ARLSTM without custom loss (0.73 °C). Surprisingly, the corrected MAE results for the baseline methods demonstrate a slight improvement for experiment (2) over experiment (1). Comparing subfigures (a) and (b) of the prediction plots shown in Figure 3.12 nicely demonstrates again, that the ARLSTM with custom loss effectively prevents overshooting and large ΔT_{out} values, while at the same time, it also leads to decreased ability of the model to capture strong variations.

For their purely theoretical reference model, Ruiz-Calvo et al. [10] calculated the daily average temperatures for their evaluation method, and then averaged over these daily values for each month, yielding a deviation of roughly $\Delta T_{\text{out}} = 0.1$ °C between their model output and the true values. In accordance with this procedure, we also calculated the daily T_{out} averages for each day of our windows, corresponding to 7 or 14 daily values, before averaging them to get a mean predicted outlet temperature and a mean true outlet temperature for each window. Contrary to their study, we did not just use one single time window for our research, but considered the whole 11-year time range of the dataset instead. Thus, we have multiple windows for both forecast scenarios. To ensure a fair comparison, we calculate ΔT for each window and then average over the windows and the folds to get a single value for each of our models, yielding:

$$\begin{aligned} \text{ARLSTM with custom loss: } \Delta T_{\text{out,1week}} &= 0.35°\text{C} \\ \text{ARLSTM without custom loss: } \Delta T_{\text{out,1week}} &= 0.37°\text{C} \\ \text{ARLSTM with custom loss: } \Delta T_{\text{out,2weeks}} &= 0.51°\text{C} \\ \text{ARLSTM without custom loss: } \Delta T_{\text{out,2weeks}} &= 0.48°\text{C} \end{aligned} \tag{3.17}$$

We have to point out that our approach included smaller forecasting windows than the reference model, but the results indicate that our proposed model could indeed achieve results comparable to those of the complex theoretical reference model. Especially, since our dataset extends over 11 years and not just one specific month, we consider the predictive capabilities of our proposed ARLSTM as sufficient. Note, however, that the averaging procedure may skew the results. Nevertheless, we had to follow this evaluation methodology to allow for at least some kind of comparison with the results of our reference model.

3.5.1. Limitations and Future Work

A key challenge of the analysis of our models' performances was to find a meaningful metric that captures the strengths and weaknesses of the predictive capabilities of our models. Initially, we aimed to use R2 scores as overall evaluation metrics, which is also the reason for the preliminary imputation check being evaluated with this metric. However, this score leads to non-trivial issues regarding the averaging procedure. Since our dataset is given in the form of multiple, mostly overlapping windows of specific sizes, the R2 scores calculated for each window may vary significantly due to slight shifts in the window positions. This inconsistency complicates the process of averaging the scores, as it yields completely different results for either calculating the score for each window separately and then averaging the scores over the windows, or concatenating all the y_{true} and y_{pred} values and calculate the R2 score just once for each fold. In addition, averaging over the folds again could skew the results. From this follows that an R2 score does not necessarily provide a consistent measure of the overall model performance. Therefore, we decided to consider MAE as our overall metric instead, which does not depend on the averaging procedure.

Regarding the imputations, in future work, further experiments with different imputation strategies may be conducted, such as defining a larger or smaller minimum length of consecutive nan entries for imputation (x), using a more sophisticated neural network, or a completely different approach. Since the imputation had such a large influence on the model's performance, this may yield valuable insights.

Another compelling research question involves the examination of performance changes for different training sizes, i.e., the number of windows used in the training set. In other words, one could investigate how many windows are required for the model to converge and to achieve satisfactory results.

4. PINNs for Pt-Ni Nanoclusters

In this part of the thesis we aim to establish a physics-informed neural network model that provides DFT energy predictions for Pt-Ni clusters with diameters in the nanometer regime. We start with a brief overview of the properties of metallic nanoclusters and their relevance in chemistry, before introducing neural network-based models for PES calculations. Finally, we delve into the practical implementation (section 4.3) of our proposed PINNs and the selected evaluation methods. Then, after presenting the results in section 4.4, we conclude the chapter with a comprehensive discussion of our findings.

4.1. Pt-Ni Nanoclusters

Nanoclusters are finite-sized quantum systems, occupying the middle ground between metallic few-atom systems (of molecular character) and the bulk. For small molecules, typically DFT methods are employed, and provide a fairly accurate description of the system. For large systems of e.g. $10^2 - 10^3$ atoms, however, DFT calculations are no longer feasible due to unfortunate scaling with system size. Therefore, other models based on several approximations must be applied to explore the complicated PES structure [60].

We opt for Pt-Ni clusters as our point of interest because for multiple reasons: To begin with, extensive previous studies are available for these systems in various mixing conditions, thereby offering a reasonable basis for comparison with our approach [61, 62, 63]. Furthermore, Pt-Ni alloys belong to the most important metal-based catalysts, owing to their high reaction activity, robustness against pollution and fine-tuning possibilities. Particularly for oxygen reduction reactions, specific structures of Pt-Ni nanoclusters exhibit some of the highest reaction activities ever observed [64]. In addition, Pt-Ni nanoclusters display interesting structural properties, since the geometrical arrangement of the atoms strongly deviates from the expected behavior. While one anticipate that the Ni atoms are pushed towards the surface because Ni has lower surface energy and Pt has higher cohesive energy, in fact the behavior is significantly affected by influences from the cluster size, composition, and temperature of the system. This makes predictions via many-body potentials rather difficult to realize [60].

4.2. Neural Networks for PES Calculations

Numerous ML methods have been established to determine the PES of many-particle systems. Most commonly, they aim to learn energy-structure relationships directly from a data set of points calculated via expensive computational chemistry methods at a sufficiently high level of theory. For sensible results, it is vital that these approximations

fulfill at least some of the most important properties of the exact PES, such as invariance with respect to transformations (rotation, translation, permutations), smoothness of the energy surface, or the capability to generalize to systems of varying size. In order to encode invariance into machine learning models, several approaches have been suggested. Most applications in computational chemistry require exact reproduction of the invariance. This can be vividly illustrated by the example of a molecular dynamics simulation, where employing a model that lacks translational and rotational invariance would yield varying energy predictions as the molecule propagates in time. This contradicts the conservation of energy and is therefore not a viable approach. A suitable method that reproduces exact invariance is to create a transformation-invariant representation of the input data. This representation can either be achieved in a preprocessing step, or as an integrated part of the model (e.g. by convolutional neural networks). A simple preprocessing approach to represent molecular geometries in an invariant way is to express the input features in terms of bond distances, bond angles, or dihedral angles [60].
To obtain the total energy E of the PES, many commonly-used ML models sum over N atomic energy contributions ϵ_i

$$E = \sum_{i=1}^{N} \epsilon_i, \tag{4.1}$$

which effortlessly incorporates the desired ability to generalize over systems of varying size. This approach supposes that the atomic energies can be expressed as functions of (a feature vector of) their local environment, which is justified by the principle of nearsightedness in quantum chemistry. The local environment of an atom i is determined by a cutoff radius R_{cut}, and the interatomic feature vectors R_{ij} vanish at the cutoff radius to avoid discontinuities.

4.2.1. Embedded Atom Model (EAM) Potentials

In the embedded atom model (EAM), which is a common choice for mixed-metal clusters, the atomic contributions to the total energy of the PES are given as

$$E_i = \frac{1}{2} \sum_{j \neq i}^{N} \phi^{\alpha\beta}(r_{ij}) + F^{\alpha}\left(\sum_{j \neq i}^{N} \rho^{\alpha\beta}(r_{ij}) \right), \tag{4.2}$$

where the first term represents the pair interactions between atoms i and j with $\phi^{\alpha\beta}(r_{ij})$ representing a repulsive pair potential, F^{α} is a so-called embedding function that accounts for the energetic cost of placing an atom in the electron density local electron density around atom i, and $\rho^{\alpha\beta}(r_{ij})$ is the pair-wise electron density function. α and β denote the element type of the corresponding atoms i and b [60]. Further extensions of this original approach, which only depends on pair-wise distances, are given by the inclusion of angular dependencies (modified embedded atom model [65]), or the incorporation of polarization effects [66]. Within this thesis, however, we will base our model on physically motivated EAM potentials given as functions of the pair-wise absolute distances between the atoms r_{ij}, and then successively replace one or more of the three terms in Equation 4.2 by neural network (NN) expressions

(i) Embedding Function: $F^{\alpha}(\rho_{\text{total}}) = \text{NN}^{F}(\rho_{\text{total}})$

(ii) Pair-wise electron density: $\rho^{\alpha\beta}(r_{ij}) = \text{NN}^{\rho}(r_{ij})$

(iii) Pair Potential: $\phi^{\alpha\beta} = \text{NN}^{\phi}(r_{ij})$

where we expressed the total electron density as

$$\rho_{\text{total}} = \sum_{j\neq i}^{N} \rho^{\alpha\beta}(r_{ij}). \tag{4.3}$$

4.2.2. SMATB Potentials

A well-known type of EAM potential is the second-moment approximation to tight binding (SMATB) [67] potential. For this potential, the embedding function is

$$F^{\alpha}(\rho) = -\sqrt{\rho}, \tag{4.4}$$

hence, it does not incorporate any atom-specific parameters. Further, the pair potential is given as

$$\phi^{\alpha\beta}(r_{ij}) = A^{\alpha\beta} e^{-p^{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}}-1\right)}, \tag{4.5}$$

and the pair-wise electron density is given as

$$\rho^{\alpha\beta}(r_{ij}) = \left(\xi^{\alpha\beta} e^{-p^{\alpha\beta}\left(\frac{r_{ij}}{r_0^{\alpha\beta}}-1\right)}\right)^2. \tag{4.6}$$

For the SMATB model in our specific system, experimental values for the equilibrium distances of the nearest neighbors in the bulk lattice can be used [68], [69]:

$$r_0 = 2.77\text{Å for Pt-Pt,}$$
$$r_0 = 2.49\text{Å for Ni-Ni, and}$$
$$r_0 = 2.63\text{Å for Pt-Ni.}$$

To avoid excessive computational costs, the pair-wise interactions are cut off at a distance $b^{\alpha\beta}$. To ensure a smooth fade-out of the interactions to zero over an interval $[a^{\alpha\beta}, b^{\alpha\beta}]$ a fifth-order polynomial is used, in accordance with [60]. The remaining parameters are taken from a standard table from Cheng et al. [70], as presented in Table 4.1.

4.3. Implementation

The overall objective of this part of the thesis is to approximate DFT calculations for the energy $E(\vec{x})$ and the gradient of the energy $\nabla E(\vec{x})$ of Pt-Ni nanoclusters of different sizes (number of atoms) and structures with physics-informed neural networks. For this purpose, in the first step, a global geometry optimization (subsection 4.3.1) needs to be

Table 4.1.: SMATB parameters for Pt-Ni, taken from [70]

α	β	A (eV)	ξ (eV)	p (eV)	q (eV)
Pt	Pt	0.1602	2.1855	13.00	3.13
Pt	Ni	0.1346	2.3338	14.838	3.036
Ni	Ni	0.0845	1.405	11.73	1.93

performed, followed by an optional sampling procedure (subsection 4.3.2) to eventually obtain a suitable training set (see subsection 4.3.3). Then, DFT calculations can be carried out for the training data to generate our reference data, following the description in subsection 4.3.4. Subsequently, as a first model the EAM method with SMATB potentials are employed, but with the parameter values refitted to the DFT data. The implementation details are provided in subsection 4.3.5. Finally, parts of the physics-based model are replaced by neural network expressions to establish efficient models for the prediction of the energies and forces of the cluster geometries. Our approach is explained in subsection 4.3.6.

4.3.1. Global Geometry Optimization

The starting point of this study is an inexpensive global geometry optimization, executed to pre-select cluster geometries for three different sizes (38, 55, 147 atoms) and varying compositions of Pt and Ni. We employ a type of basin hopping algorithm, suggested by Rossi and Ferrando [47] and evaluate the corresponding energies and forces with SMATB potentials. Mixture ratios of roughly 1:3, 1:1 and 1-element-impurities are chosen for each cluster size. The starting geometries for the $N = 38$ clusters are truncated octahedra, while for the $N = 55$ and $N = 47$ clusters, they are icosahedra. The optimization starts with Pt and Ni atoms randomly distributed throughout the structure (random mixing). For each 1:1 mixture, in addition, a so-called "janus" configuration is used as starting configuration for the optimization runs. In this configuration, the clusters are basically unmixed and consist of two halves, containing only Pt and Ni atoms, respectively [60].

4.3.2. Sampling Procedure

Typically, the initial dataset comprises a large number of cluster geometries, specified by the atomic positions. From this large dataset, it is often required to sample for each cluster size a number of n data points that serves as the base for our DFT calculations. The value of n should be chosen as a reasonable trade-off between computational resource requirements for the DFT calculations and sufficient samples to train the neural networks. To obtain a suitable training set, it is vital to ensure that the sub-sampled data points contain all major geometry types. To this end, a sampling strategy similar to Roncaglia & Ferrando [71] can be employed, where first a descriptor of the original data is created in the form of a Coulomb matrix, from which the eigenvalues can be calculated. The eigenvalues can then be used as input for a k-medoids clustering algorithm (with the *sklearn*-implementation). To save computational resources, some preprocessing steps should be applied prior to the clustering, given by: standard-scaling, PCA dimensionality reduction to 50 components, and t-SNE dimensionality reduction to 2 components (motivated by visualization purposes). It is obvious to define $k = n$ for the k-medoids algorithm and take the data points representing the cluster centers (medoids) as a new sub-set for the subsequent calculations. The results of this sampling technique applied to an exemplary dataset of 55-atom Pt-Ni clusters, are represented in Figure 4.1.

4.3.3. Training Set Selection

In our specific case, the initial dataset generated with the global optimization in subsection 4.3.1 contains a total of 1463 cluster geometries, out of which we use the 110 data points that correspond to the geometries with the lowest energies (5 for each composition) as test set, and split the remaining data points into 90% training and 10% validation set. Since we do not have an extensive initial dataset for this study, we refrain from applying the sampling method introduced in subsection 4.3.2 but use all available data points instead. This dataset serves as a basis for the creation of the purely physics-driven EAM model with SMATB potentials, our proposed variations of PINNs, and the DFT data we use as reference, i.e. to fit our models.

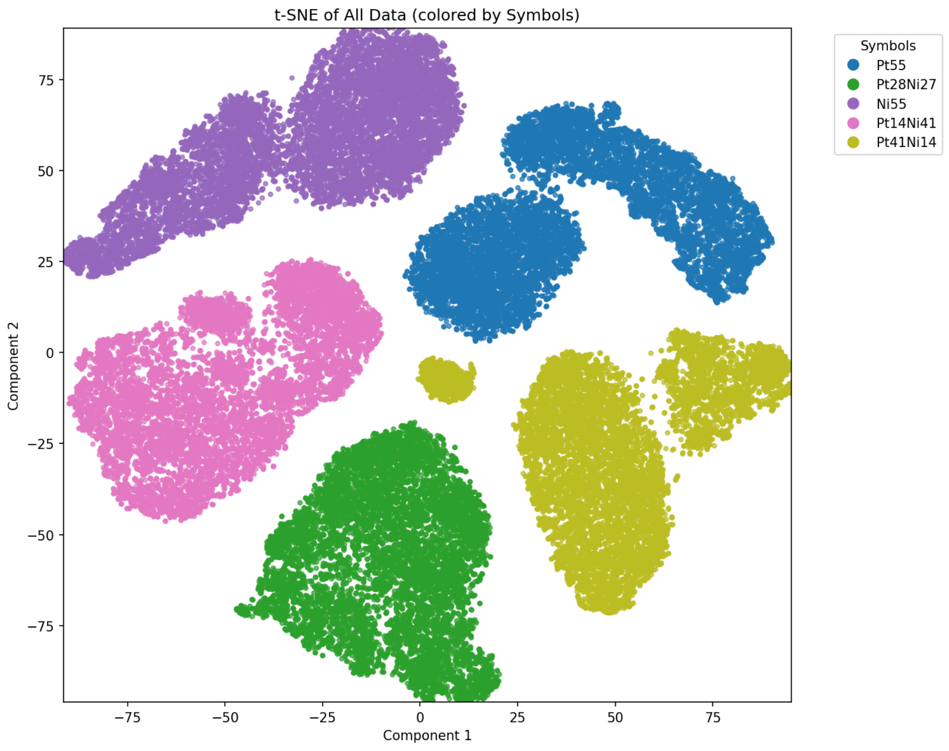
4.3.4. DFT Training Data Generation

To generate DFT data from the globally optimized geometries of the initial data set, the software package *Quantum Espresso*[53] is used. We configure a supercell as a cube with edge length of 45 bohr and run the unrestricted DFT calculations using a combination of PBE functionals and ultrasoft pseudopotentials. The wave function cutoff is set to 40 Ry and the density cutoff to 400 Ry. In addition, the Gamma point is used for sampling the Brillouin zone, and a smearing method introduced by Methfessel and Paxton [72] is applied with a value of 0.002 Ry.

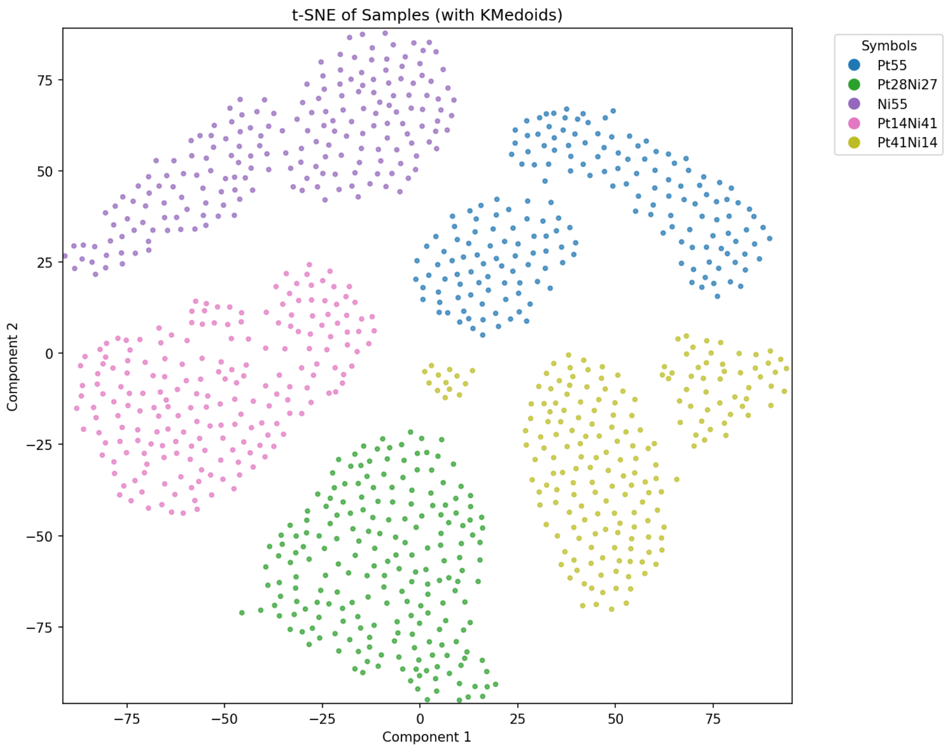
To carry out the DFT calculations, an input file with the above specified parameters is created and submitted the to quantum espresso package. An example of such an input file is in Appendix A). Since the DFT calculations provide the total energy of all valence electrons $E_{\text{tot}}^{\text{DFT}}$, whereas the SMATB method only predicts the bonding energy of the cluster $E_{\text{bond}}^{\text{SMATB}}$, we need to conduct additional DFT calculations for the isolated atoms in order to obtain the reference DFT bonding energy $E_{\text{ref}}^{\text{DFT}}$ as follows [60]:

$$E_{\text{ref}}^{\text{DFT}} = E_{\text{tot}}^{\text{DFT}} - N_{\text{Pt}} E_{\text{Pt}}^{\text{DFT}} - N_{\text{Ni}} E_{\text{Ni}}^{\text{DFT}}. \tag{4.7}$$

In addition, we also use the DFT predictions for the atomic forces in order to increase the number of data points.



(a) All data points after preprocessing



(b) Selected samples with K-medoids after preprocessing

Figure 4.1.: Sampling results for 55-atom clusters

4.3.5. Refitted SMATB Model for Approximations of DFT Calculations

The refitted SMATB model will be constructed in a way that it yields two outputs: the atomic energies E_k^{model} and the components l of the atomic forces F_{kl}^{model}. These outputs are given as:

$$\begin{aligned} E_k^{\text{model}} &= \sum_i E_{k,i}^{\text{model}} \\ \mathbf{F}_k^{\text{model}} &= -\nabla E_k^{\text{model}}, \end{aligned} \tag{4.8}$$

where the index k represents the sample and i gives the index of the atom of the corresponding cluster sample.

Instead of using the parameter from Table 4.1, we refit them to the reference DFT data (Equation 4.7) in order to add more flexibility to the model. To this end, we use the loss function

$$\mathcal{L} = w_e \frac{1}{M} \sum_k^M \frac{1}{N_k} (E_k^{\text{model}} - E_k^{\text{DFT}})^2 + w_f \frac{1}{M} \sum_k^M \frac{1}{3N_k} \sum_l^{3N_k} (F_{kl}^{\text{model}} - F_{kl}^{\text{DFT}})^2 + \lambda \sum_k^{N_k} w_k^2, \tag{4.9}$$

where w_e and w_f are weighting factors that determine the influence of the mean squared errors of the energies respectively forces, and the last term represents a l2-regularization term applied to the model's weights and specified by the hyperparameter λ. To simplify the implementation, we multiply the pair density $\rho^{\alpha\beta}$ and the pair potential $\phi^{\alpha\beta}$ by a cutoff function, instead of fitting a fifth-order polynomial for fading out the long-range interactions [60].

$$f_{\text{cut}}^{\alpha\beta}(r) = \begin{cases} 1, & \text{for } r \leq a^{\alpha\beta} \\ 1 - 10\hat{r}^3 + 15\hat{r}^4 - 6\hat{r}^5, & \text{for } a^{\alpha\beta} < r < b^{\alpha\beta} \\ 0, & \text{for } r \geq b^{\alpha\beta}, \end{cases} \tag{4.10}$$

where $\hat{r} = \frac{r - a^{\alpha\beta}}{b^{\alpha\beta} - a^{\alpha\beta}}$ is the scaled distance. The model is implemented using the TensorFlow [56] library with the hyperparameters listed in Table 4.2.

4.3.6. PINNs for Approximating DFT Calculations

In contrast to the PINN we implemented for the borehole output prediction in chapter 3, we will now set up the approach from the other direction: Starting from the purely physics-based model of the Pt-Ni nanocluster PES in form of the SMATB method, we will gradually replace terms of the Equation 4.2 by neural network expressions.

PINN for Embedding Function

The first part in Equation 4.2 we aim to replace is the embedding function. To do so, we create a small TensorFlow neural network with tanh activation functions and l2 regularizer. The inputs is given by the density function ρ. The network then outputs

Hyperparameter	values
optimizer	Adam
learning rate	0.001
max. epochs	500
early stopping patience	20
batch size	20
w_f	1
w_e	1
λ	0

Table 4.2.: Hyperparameters used for refitting the SMATB parameters

the embedding for this specific density. To ensure that the physical constraint for the embedding

$$F^{\alpha}(\rho = 0) = 0 \tag{4.11}$$

is fulfilled, we add this constraint as an additional term to the loss function, yielding:

$$\begin{aligned}\mathcal{L} = w_e \frac{1}{M}\sum_{k}^{M} \frac{1}{N_k}(E_k^{\text{model}} - E_k^{\text{DFT}})^2 + w_f \frac{1}{M}\sum_{k}^{M} \frac{1}{3N_k}\sum_{l}^{3N_k}(F_{kl}^{\text{model}} - F_{kl}^{\text{DFT}})^2 \\ + w_b \frac{1}{M}\sum_{k}^{M}(F_k^{\alpha}(0))^2 + \lambda \sum_{k}^{N_k} w_k^2\end{aligned} \tag{4.12}$$

where $F_k^{\alpha}(0)$ is the embedding function at a density of zero, and w_b is a weighting factor for the influence of the embedding constraint. The remaining terms of Equation 4.2 are left unchanged from the SMATB refit model introduced in subsection 4.3.5 to allow for maximal flexibility. To optimize the model performance, we implement a small grid search to tune the hyperparameter b. The best values for the hidden layer sizes and regularization parameter λ were taken from [60]. The total hyperparameter space is given below in Table 4.3. The results of the fitted model for the best hyperparameter found are presented in Table 4.7.

PINN for Pair Density Function

In this section, we replace the pair density function in Equation 4.9 by a neural network. Again, we implement the networks separately for the two atom types in the clusters and use tanh activation functions, l2 regularization. The pair-wise distances r_{ij} serve as input for the network, which outputs the pair density $\rho^{\alpha\beta}(r_{ij})$. To account for fundamental physical properties, we ensure that

$$\rho^{\alpha\beta}(r_{ij}) \geq 0, \tag{4.13}$$

Hyperparameter	**values**
optimizer	Adam
learning rate	0.001
max. epochs	500
early stopping patience	20
batch size	20
w_f	1
w_e	1
w_b	$[10, 45, 100]$
λ	10^{-5}
hidden layers sizes	[15, 15]

Table 4.3.: Hyperparameter space used for grid search of PINN for embedding function

by adding this constraint to the loss function through a step-function, and weight the term with the parameter w_c:

$$\begin{aligned}\mathcal{L} = w_e \frac{1}{M}\sum_k^M \frac{1}{N_k}(E_k^{\text{model}} - E_k^{\text{DFT}})^2 + w_f \frac{1}{M}\sum_k^M \frac{1}{3N_k}\sum_l^{3N_k}(F_{kl}^{\text{model}} - F_{kl}^{\text{DFT}})^2 \\ + w_c \frac{1}{M}\sum_k^M \rho_{\text{penalty},k} + \lambda \sum_k^{N_k} w_k^2,\end{aligned} \tag{4.14}$$

where

$$\rho_{\text{penalty},k} = \begin{cases} |\rho^{\alpha\beta}(r_{ij})|, & \text{for } \rho^{\alpha\beta}(r_{ij}) < 0 \\ 0, & \text{else.} \end{cases} \tag{4.15}$$

To optimize the results, we perform a small grid search on the hyperparameter w_c. The best values for the hidden layer sizes and regularization parameter λ were taken from [60]. The corresponding hyperparameter space for all employed parameters is given below in Table 4.4.

The results of the best hyperparameters found from applying the fitted model to the test set, are presented in Table 4.7.

PINN for both embedding function and pair density function

Now we replace both the embedding function and the pair density function by neural network expressions and add the corresponding physical constraints to the loss function, yielding:

$$\begin{aligned}\mathcal{L} = w_e \frac{1}{M}\sum_k^M \frac{1}{N_k}(E_k^{\text{model}} - E_k^{\text{DFT}})^2 + w_f \frac{1}{M}\sum_k^M \frac{1}{3N_k}\sum_l^{3N_k}(F_{kl}^{\text{model}} - F_{kl}^{\text{DFT}})^2 \\ + w_b \frac{1}{M}\sum_k^M (F_k^{\alpha}(0))^2 + w_c \frac{1}{M}\sum_k^M \rho_{\text{penalty},k} + \lambda \sum_k^{N_k} w_k^2.\end{aligned} \tag{4.16}$$

Hyperparameter	**values**
optimizer	Adam
learning rate	0.001
max. epochs	500
early stopping patience	20
batch size	20
w_f	1
w_e	1
w_c	$[500, 10000, 20000]$
λ	10^{-6}
hidden layer sizes	[15, 15, 15]

Table 4.4.: Hyperparameter space used for grid search of PINN for pair density function

Again, we perform a small grid search to tune the hyperparameters w_b and w_c. The best values for the hidden layer sizes and regularization parameter λ were taken from [60]. The overall hyperparameter space is represented in Table 4.5. The corresponding results

Hyperparameter	**values**
optimizer	Adam
learning rate	0.001
max. epochs	500
early stopping patience	20
batch size	20
w_f	1
w_e	1
w_b	$[10, 45, 100]$
w_c	$[500, 20000, 50000]$
λ	10^{-6}
hidden layer sizes	[20, 20, 20]

Table 4.5.: Hyperparameter space used for grid search of PINN for both embedding and pair density function

are represented in Table 4.7.

NNs without constraints for both Embedding Function and Pair Density Function

To get a feeling for the influence of the physical constraints on the performance of the PINNs, we also implement the model where we replace both the embedding function and the pair density function without the additional constraint terms in the loss function, i.e.

$$\mathcal{L} = w_e \frac{1}{M} \sum_k^M \frac{1}{N_k} (E_k^{\text{model}} - E_k^{\text{DFT}})^2 + w_f \frac{1}{M} \sum_k^M \frac{1}{3N_k} \sum_l^{3N_k} (F_{kl}^{\text{model}} - F_{kl}^{\text{DFT}})^2 + \lambda \sum_k^{N_k} w_k^2. \tag{4.17}$$

We perform the same grid search as for the model introduced in subsubsection 4.3.6, and use the hyperparameters given in Table 4.6. The results for applying the model with

Hyperparameter	**values**
optimizer	Adam
learning rate	0.001
max. epochs	500
batch size	20
w_f	1
w_e	1
λ	10^{-6}
hidden layer sizes	[20, 20, 20]

Table 4.6.: Hyperparameter space used for grid search of PINN for both embedding and pair density function without additional physical constraints

the best hyperparameters to the test set are represented in Table 4.7.

4.3.7. Evaluation Method

To evaluate and compare the performance of our employed models, we use RMSE as our overall metric. We split the data into 90% training set and 10% validation set, in addition to our test set which contains 110 samples, as defined in subsection 4.3.3. After creating and fitting the models according to section 4.3, we apply them to the test set and calculate the corresponding evaluation metric, i.e. RMSE. The results of our evaluation are presented in section 4.4.

4.4. Results

In the Table 4.7 below we present the results of applying our models to the test data. The best hyperparameters found from our small grid search are:

- $b = 100$, for the embedding PINN with constraint
- $c = 20000$, for the pair density PINN with constraint, and
- $b = 100, c = 500$ for the PINN replacing both embedding and density function with constraints.

Table 4.7.: Results for the validation (val.) set and test set for employing PINNs for approximated DFT calculations with the RMSE values for the energies E in meV/atom and the forces F in meV/Å. The results of the SMATB approach with the parameters from the standard table are taken from the study of R. Meyer [60].

Model	**Val. Set** E	**Val. Set** F	**Test Set** E	**Test Set** F
SMATB	844.6	336.6	830.2	336.2
SMATB refit	43.3	197.3	68.7	211.8
PINN for $F^\alpha(\rho)$	43.1	198.5	59.9	212.1
PINN for $\rho^{\alpha\beta}(r_{ij})$	77.0	293.0	115.6	277.8
PINN for $F^\alpha(\rho)$ and $\rho^{\alpha\beta}(r_{ij})$	76.4	212.3	96.9	211.9
NN for $F^\alpha(\rho)$ and $\rho^{\alpha\beta}(r_{ij})$	150.1	327.4	215.4	277.7

An exemplary loss curve for the PINN for the embedding function is given in Figure 4.2 and a discussion is provided in section 4.5.

4.5. Discussion

The results in Table 4.7 show that a simple refit of the parameters of the SMATB potential to DFT data already yields a drastic decrease in the root mean squared errors. In particular, the RMSE on the test set shrinks from 839.2 meV/atom to 68.7 meV/atom for the energies and from 336.2 meV/Å to 211.8 meV/Å for the forces. By employing a PINN for the embedding function of the SMATB model, the errors on the test set can be further reduced to 59.5 meV/atom for the energies, but slightly increase to 212.1 meV/Å for the forces. The PINN for the pair density function, however, does not yield better results than the simple SMATB refit. The root mean squared errors for this model are 115.6 meV/atom for the energies and 277.8 meV/Å for the forces. The model where both the embedding and density function are replaced by PINNs performs better, with an RMSE for the energies of 96.9 eV/atom and 211.9 meV/Å for the forces, but still worse than the SMATB refit model. This could be expected since the model is a

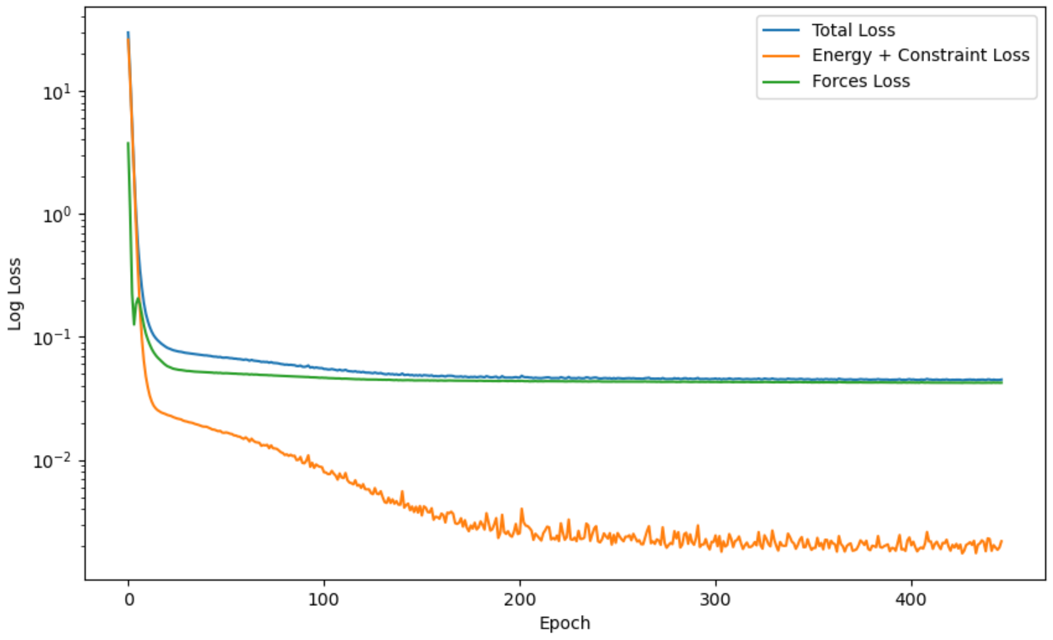


Figure 4.2.: Logarithmic loss curve of the PINN for the embedding function over epochs

mix of the PINN for embedding, which outperforms the SMATB refit, and the PINN for the pair density, which performed significantly worse than the SMATB refit model. Finally, the neural network that replaces both the embedding function and pair density function without physical constraints performs worse than all the other models except for the standard SMATB model. This verifies our assumption that physical constraints are advantageous in terms of guiding the model toward physically consistent solutions.

The root mean squared errors on the validation set show similar tendencies as the errors on the test set. What stands out though, is that even though the SMATB refit yields very similar results on the validation set as the PINN for the embedding function, it performs significantly worse than the PINN for the energies of the test set.

By observing the exemplary plot of the loss curve (see Figure 4.2), one can see that the overall convergence behavior of the model looks as expected, with all loss terms continuously decreasing over time, even though there are some slight fluctuations in the combined energy and constraints loss towards greater epoch numbers. Furthermore, the graph shows a little "shoulder" in the forces loss term after approximately 2-5 epochs. This behavior is caused by the fact that we started the training of the model from already relaxed, pre-optimized structures. Therefore, the network tends to predict forces near zero at the beginning of the optimization to minimize the RMSE values. However, to further decrease the energy loss, the forces need to be adjusted, which temporarily causes a slight increase in the forces loss term, before it eventually trends back towards lower values.

5. Conclusion

Within the course of this thesis, we examined the capabilities of physics-informed neural networks in the realm of two different practical applications. Our overarching goal was to investigate whether these hybrid approaches, merging the rigor of physically motivated models with the flexibility of data-driven methods, can leverage the performance of conventional techniques.

In the context of borehole output predictions for optimizing the energy efficiency of ground-source heat pumps, we found that, indeed, adding simple physical constraints to the loss function of a data-driven model can enhance its predictive capabilities and decrease the mean absolute error from 0.60 °C to 0.53 °C for a one-week-forecast, and from 0.73 °C to 0.70 °C for a two-week-forecast. Furthermore, our study indicates that the performance of our model is not drastically below the performance of a highly sophisticated and computationally costly theoretical reference model.

Our study on PINNs for PES calculations of Pt-Ni nanoclusters revealed that resource-intensive DFT calculations can be fairly well approximated by inexpensive models if parts of the models are replaced by neural network expressions with additional physics-based constraints. We found a model derived from the SMATB approach with replacement of the embedding function by a neural network with additional physics-based constraints to work best, achieving a RMSE on the test of 59.9 eV/atom for the energies and 212.1 meV/Å for the forces, while, in comparison, the purely theoretical embedded atom model approach deviated by 830.2 eV/atom and 336.2 meV/Å, respectively, from the reference DFT data. This illustrates that costly DFT calculations can indeed be approximated to a large extent by comparatively efficient PINNs.

To conclude, this work demonstrated that PINNs have great potential to enhance the accuracy and efficiency of complex predictive models across different fields. By effectively integrating physics-based constraints into data-driven neural networks, we showed that PINNs can boost the efficiency of ground-source heat pumps through improved predictions of the temporal behavior of borehole heat exhangers, and are also capable of predicting energy-structure relationships for Pt-Ni nanoclusters. These findings suggest that PINNs can be considered a powerful tool that provides a reasonable trade-off between computational costs and high prediction accuracy. Future research might explore further applications of PINNs, potentially leading to breakthroughs in other areas where high computational demands are currently limiting the use of sophisticated modeling techniques.

A. Appendix A

This appendix presents an exemplary input file for DFT calculations with Quantum Espresso.

```
&CONTROL
  calculation  = 'scf',
  tprnfor      = .true.,
  prefix       = 'Pt19Ni19_test01',
  disk_io      = 'none',
  pseudo_dir   = '/home/lv71054/rmeyer/pseudo/',
  outdir       = '/home/lv71054/rmeyer/.tempdir/PtNi/dataset/Pt19Ni19_test01/'
/
&SYSTEM
  ibrav       = 1,
  celldm(1)   = 45.0
  nat         = 38,
  ntyp        = 2,
  ecutwfc     = 40.D0,
  ecutrho     = 400.D0,
  occupations = 'smearing',
  smearing    = 'mp',
  degauss     = 0.002,
  nspin       = 2,
  starting_magnetization(1) = 0.9,
  starting_magnetization(2) = 0.3
/
&ELECTRONS
  electron_maxstep = 300,
  diago_david_ndim = 2,
  mixing_mode      = 'local-TF',
  mixing_beta      = 0.1D0,
/

ATOMIC_SPECIES
Ni 58.6934 Ni.pbe-n-rrkjus_psl.0.1.UPF
Pt 195.084 Pt.pbe-n-rrkjus_psl.0.1.UPF
ATOMIC_POSITIONS {angstrom}
Pt     18.5419389    -20.0481783    -19.5678484
Pt      19.9690325    -16.9903804    -15.7368394
Pt      21.6533501    -21.7260545    -15.1146977
Pt      21.4736331    -17.7653048    -13.6356758
Pt      20.1973837    -18.9904737     -11.601925
Pt      23.0747711    -19.3002012    -15.2527828
Pt      16.6919253    -22.6688926    -15.7489602
Pt      19.6065356    -20.9316546    -13.4255888
Pt      19.9012611    -17.7027332    -19.6271091
Pt      18.2850412    -15.9248275    -13.9171296
Pt      16.3876942    -15.4445854    -15.8436709
Pt      16.5949444    -19.0852304    -17.6700514
Pt      14.9827023    -17.8751922    -15.7051634
Pt      21.6291543    -18.4056732    -17.5059157
Pt      16.4493167    -19.1114463    -13.7045833
Pt      17.5901473    -17.4864246    -11.8325365
Pt      18.0881059    -20.2256873    -15.5759128
Pt      20.0743198    -21.1030889    -17.3606414
Pt      18.1623707    -16.4151497    -17.8117333
Ni      21.9983837    -20.1762757    -13.0903954
Ni      22.5803522    -20.7492639    -17.2699952
Ni      15.5397239    -20.3925831    -15.7322527
Ni      18.9580418    -18.4411802     -13.769106
Ni       21.061672    -19.9655018    -19.4146474
Ni      20.5558862    -19.4726069    -15.3879423
Ni      17.4982907    -17.7085383    -15.6545589
Ni      20.7158194    -16.0241846     -17.934607
Ni       15.880008    -16.6465609     -13.630308
Ni      19.2150329    -22.5171468    -15.4023497
Ni      17.1419162    -21.5592387    -13.5723639
Ni      17.3553141    -17.8258643    -19.7500602
Ni      15.6446752    -16.7472481    -17.8846826
Ni      19.1087911    -18.7487205    -17.4438413
Ni      22.5347459    -16.8472984    -15.7435016
Ni      17.8894115    -19.9880939      -11.77642
Ni      19.9154132    -16.5219471    -12.0719328
Ni      17.5575582    -21.4710212    -17.7559988
Ni      18.8282576    -14.7134492    -16.0739665
K_POINTS Gamma
```

Bibliography

[1] G. E. Karniadakis et al. *Physics-informed machine learning.* June 2021. DOI: 10.1038/s42254-021-00314-5.

[2] S. Cuomo et al. "Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next". In: *Journal of Scientific Computing* 92 (Sept. 2022). ISSN: 15737691. DOI: 10.1007/s10915-022-01939-z.

[3] K. Kashinath et al. *Physics-informed machine learning: Case studies for weather and climate modelling.* Apr. 2021. DOI: 10.1098/rsta.2020.0093.

[4] Y. Bai, T. Chaolu, and S. Bilige. "The application of improved physics-informed neural network (IPINN) method in finance". In: *Nonlinear Dynamics* 107.4 (Mar. 2022), pp. 3655–3667. ISSN: 1573269X. DOI: 10.1007/s11071-021-07146-z.

[5] P. Nejat et al. *A global review of energy consumption, CO2 emissions and policy in the residential sector (with an overview of the top ten CO2 emitting countries).* 2015. DOI: 10.1016/j.rser.2014.11.066.

[6] S. Younger. *NASA Analysis Confirms a Year of Monthly Temperature Records.* June 2024. URL: https://www.nasa.gov/earth/nasa-analysis-confirms-a-year-of-monthly-temperature-records/.

[7] S. Javed. "Thermal modelling and evaluation of borehole heat transfer". PhD thesis. Göteborg: Chalmers University of Technology, 2012. ISBN: 9789173856232.

[8] N. Ahmed et al. *Optimal design, operational controls, and data-driven machine learning in sustainable borehole heat exchanger coupled heat pumps: Key implementation challenges and advancement opportunities.* June 2023. DOI: 10.1016/j.esd.2023.04.004.

[9] F. Ruiz-Calvo et al. "Reference data sets for validating and analyzing GSHP systems based on an eleven-year operation period". In: *Geothermics* 64 (Nov. 2016), pp. 538–550. ISSN: 03756505. DOI: 10.1016/j.geothermics.2016.08.004.

[10] F. Ruiz-Calvo et al. "Coupling short-term (B2G model) and long-term (g-function) models for ground source heat exchanger simulation in TRNSYS. Application in a real installation". In: *Applied Thermal Engineering* 102 (June 2016), pp. 720–732. ISSN: 13594311. DOI: 10.1016/j.applthermaleng.2016.03.127.

[11] H. Zhen et al. "Physically Compatible Machine Learning Study on the Pt-Ni Nanoclusters". In: *Journal of Physical Chemistry Letters* 12.5 (Feb. 2021), pp. 1573–1580. ISSN: 19487185. DOI: 10.1021/acs.jpclett.0c03600.

[12] J. P. Mueller and L. Massaron. *Deep Learning for Dummies.* For Dummies, 2019. URL: https://learning.oreilly.com/library/view/deep-learning-for/9781119543046/c01.xhtml#h2-1.

[13] F. Fleuret. *The Little Book of Deep Learning.* Université de Genève, 2023.

[14] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133.

[15] A. Géron. *Neural Netwroks and Deep Learning.* O'Reilly Media, Inc., 2018. URL: https://learning.oreilly.com/library/view/neural-networks-and/9781492037354/ch01.html#idm139624972225600.

[16] U. bin Waheed et al. "PINNeik: Eikonal solution using physics-informed neural networks". In: *Computers and Geosciences* 155 (Oct. 2021). ISSN: 00983004. DOI: 10.1016/j.cageo.2021.104833.

[17] H. Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (July 2019), pp. 917–963. ISSN: 1573756X. DOI: 10.1007/s10618-019-00619-1.

[18] G. V. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314. URL: https://api.semanticscholar.org/CorpusID:3958369.

[19] R. M. Schmidt. "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview". In: (Nov. 2019). URL: http://arxiv.org/abs/1912.05911.

[20] S. Hochreiter and J. Schmidhuber. "LONG SHORT-TERM MEMORY". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[21] F. A. Gers, J. Schmidhuber, and F. Cummins. "Learning to Forget: Continual Prediction with LSTM". In: DOI: 10.1049/cp:19991218.

[22] S. Mallat. *Understanding deep convolutional networks.* Apr. 2016. DOI: 10.1098/rsta.2015.0203.

[23] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. ISSN: 10902716. DOI: 10.1016/j.jcp.2018.10.045.

[24] S. Cai et al. *Physics-informed neural networks (PINNs) for fluid mechanics: a review.* Dec. 2021. DOI: 10.1007/s10409-021-01148-1.

[25] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: (Dec. 2014). URL: http://arxiv.org/abs/1412.6980.

[26] H. Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (July 2019), pp. 917–963. ISSN: 1573756X. DOI: 10.1007/s10618-019-00619-1.

[27] B. Lim and S. Zohren. *Time-series forecasting with deep learning: A survey.* Apr. 2021. DOI: 10.1098/rsta.2020.0209.

[28] S. Shrivastava. *Cross Validation in Time Series.* https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4. 2023. URL: `https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4`.

[29] I. Sarbu and C. Sebarchievici. *A comprehensive review of thermal energy storage.* Jan. 2018. DOI: `10.3390/su10010191`.

[30] *Thermal Energy Storage- Technology Brief.* Tech. rep. International Renewable Energy Agency, 2013. URL: `https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2013/IRENA-ETSAP-Tech-Brief-E17-Thermal-Energy-Storage.pdf`.

[31] S. C. Gupta. *Thermodynamics.* Pearson Education Canada, 2005. ISBN: 9788131717950. URL: `https://books.google.at/books?id=QNSv4chQWoYC`.

[32] M. Thirumaleshwar. *Fundamentals of Heat and Mass Transfer.* Always learning. Pearson Education, 2009. ISBN: 9788177585193. URL: `https://books.google.at/books?id=b2238B-AsqcC`.

[33] Wikipedia. *Wärmeleitungsgleichung – Wikipedia, Die freie Enzyklopädie.* 2023. URL: `https://de.wikipedia.org/wiki/W%C3%A4rmeleitungsgleichung`.

[34] Colorado State University. *Energy Balances.* URL: `https://www.engr.colostate.edu/CBE101/topics/energy_balances.html`.

[35] Y. A. Çengel and M. A. Boles. *Thermodynamics: An Engineering Approach.* 5th ed. McGraw-Hill Education, 2006.

[36] M. Climo et al. "The rise and rise of geothermal heat pumps in New Zealand." In: *New Zealand Geothermal Workshop 2012.* Auckland, Nov. 2012.

[37] G. Alva, Y. Lin, and G. Fang. *An overview of thermal energy storage systems.* Feb. 2018. DOI: `10.1016/j.energy.2017.12.037`.

[38] M. De Rosa et al. "Borehole modelling: A comparison between a steady-state model and a novel dynamic model in a real ON/OFF GSHP operation". In: *Journal of Physics: Conference Series.* Vol. 547. 1. Institute of Physics Publishing, 2014. DOI: `10.1088/1742-6596/547/1/012008`.

[39] Y. Guo et al. "Considering buried depth for vertical borehole heat exchangers in a borehole field with groundwater flow—An extended solution". In: *Energy and Buildings* 235 (Mar. 2021). ISSN: 03787788. DOI: `10.1016/j.enbuild.2021.110722`.

[40] K. S. Lee. *Underground Thermal Energy Storage.* Vol. 75. Springer Verlag, 2013. ISBN: 9781447142720. DOI: `10.1007/978-1-4471-4273-7`.

[41] P. Eskilson. "Thermal Analysis of Heat Extraction Boreholes". PhD thesis. Lund, Sweden: University of Lund, 1987.

[42] G. Hellström. "Ground heat storage : thermal analyses of duct storage systems". English. PhD thesis. Mathematical Physics, 1991. ISBN: 91-628-0290-9.

[43] I. N. Levine. *Quantum Chemistry.* Pearson, 2013. ISBN: ISBN-13: 9780321918185. URL: `www.pearsonhighered.com/advchemistry..`

[44] F. Jensen. *Introduction to Computational Chemistry Second Edition.* John Wiley & Sons, Ltd, 2007.

[45] P. Hadley. *Lecture Notes Solid State Physics.* URL: `https://lampz.tugraz.at/~hadley/ss1/molecules/hamiltonian.php`.

[46] A. Hauser. *Lecture Notes Modelling of Molecular Systems.* Graz, 2024.

[47] G. Rossi and R. Ferrando. “Searching for low-energy structures of nanoparticles: A comparison of different methods and algorithms”. In: *Journal of Physics Condensed Matter* 21.8 (2009). ISSN: 1361648X. DOI: `10.1088/0953-8984/21/8/084208`.

[48] R. Fletcher. *Practical Methods of Optimization.* 2nd. New York: John Wiley & Sons, 1987. ISBN: 978-0-471-91547-8.

[49] P. Hohenberg and W. Kohn. “Inhomogeneous Electron Gas”. In: *Physical Review* 136.3B (1964). DOI: `https://doi.org/10.1103/PhysRev.136.B864`.

[50] S. Di Cataldo. *A quick introduction to Quantum Espresso.* Tech. rep. Graz University of Technology, Sept. 2019. URL: `https://youtu.be/1AH2pkijDPg`.

[51] C. Lee, W. Yang, and R. G. Parr. “Development of the Colic-Salvetti correlation-energy formula into a functional of the electron density”. In: *Physical review. B, Condensed matter* 37.2 (1988), pp. 785–789. DOI: `10.1103/physrevb.37.785`.

[52] J. P. Perdew, K. Burke, and M. Ernzerhof. “Generalized Gradient Approximation Made Simple”. In: *Physical Review Letters* 77.18 (1996), pp. 3865–3868. DOI: `10.1103/PhysRevLett.77.3865`.

[53] P. Giannozzi et al. “QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials”. In: *Journal of Physics: Condensed Matter* 21.39 (2009), 395502 (19pp). URL: `http://www.quantum-espresso.org`.

[54] P. Monzó et al. *Experimental Validation of a Numerical Model for the Thermal Response of a Borehole Field.* Tech. rep.

[55] G. Ciaburro, V. K. Ayyadevara, and A. Perrier. *Hands-On Machine Learning on Google Cloud Platform.* Packt Publishing, Apr. 2018.

[56] M. Abadi et al. “Tensorflow: A system for large-scale machine learning”. In: *12th Symposium on Operating Systems Design and Implementation.* 2016, pp. 265–283.

[57] P. Eskilson. “Superposition borehole model”. In: *Manual for computer code* (1986).

[58] M. De Rosa et al. “A novel TRNSYS type for short-term borehole heat exchanger simulation: B2G model”. In: *Energy Conversion and Management* 100 (Aug. 2015), pp. 347–357. ISSN: 01968904. DOI: `10.1016/j.enconman.2015.05.021`.

[59] F. Ruiz-Calvo et al. “Experimental validation of a short-term Borehole-to-Ground (B2G) dynamic model”. In: *Applied Energy* 140 (Feb. 2015), pp. 210–223. ISSN: 03062619. DOI: `10.1016/j.apenergy.2014.12.002`.

[60] R. Meyer. “Machine Learning in Computational Chemistry”. PhD thesis. 2021.

[61] R. Ferrando, J. Jellinek, and R. L. Johnston. “Nanoalloys: From theory to applications of alloy clusters and nanoparticles”. In: *Chemical Reviews* 108.3 (2008), pp. 845–910. ISSN: 00092665. DOI: 10.1021/cr040090g.

[62] G. Wang et al. “Monte Carlo simulations of segregation in Pt-Ni catalyst nanoparticles”. In: *Journal of Chemical Physics* 122.2 (2005). ISSN: 00219606. DOI: 10.1063/1.1828033.

[63] C. Di Paola and F. Baletto. “Oxygen adsorption on small PtNi nanoalloys”. In: *Physical Chemistry Chemical Physics*. Vol. 13. 17. May 2011, pp. 7701–7707. DOI: 10.1039/c0cp01662d.

[64] V. R. Stamenkovic et al. “Improved oxygen reduction activity on Pt3Ni(111) via increased surface site availability”. In: *Science* 315.5811 (Jan. 2007), pp. 493–497. ISSN: 00368075. DOI: 10.1126/science.1135941.

[65] M. I. Baskes, J. S. Nelson, and A. F. Wright. “Semiempirical modified embedded-atom potentials for silicon and germanium”. In: *Phys. Rev. B* 40.9 (Sept. 1989), pp. 6085–6100. DOI: 10.1103/PhysRevB.40.6085. URL: https://link.aps.org/doi/10.1103/PhysRevB.40.6085.

[66] H. Bhattarai, K. E. Newman, and J. D. Gezelter. “Polarizable Potentials For Metals: The Density Readjusting Embedded Atom Method (DR-EAM)”. In: (Mar. 2019). DOI: 10.1103/PhysRevB.99.094106. URL: http://arxiv.org/abs/1904.00263%20http://dx.doi.org/10.1103/PhysRevB.99.094106.

[67] D. Tomanek, A. A. Aligia, and C. A. Balseiro. “Calculation of elastic strain and electronic effects on surface segregation”. In: *Physical Review B* 32.8 (1985), pp. 5051–5056.

[68] M. W. Finnis and J. E. Sinclair. “A simple empirical N-body potential for transition metals”. In: *Philosophical Magazine A* 50.1 (1984), pp. 45–55. DOI: 10.1080/01418618408244210. URL: https://doi.org/10.1080/01418618408244210.

[69] C. Kittel. *Introduction to Solid State Physics Charles Kittel*. 2005. ISBN: 978-1-119-45416-8.

[70] D. Cheng, S. Yuan, and R. Ferrando. “Structure, chemical ordering and thermal stability of Pt-Ni alloy nanoclusters”. In: *Journal of physics. Condensed matter : an Institute of Physics journal* 25 (Aug. 2013), p. 355008. DOI: 10.1088/0953-8984/25/35/355008.

[71] C. Roncaglia and R. Ferrando. “Machine Learning Assisted Clustering of Nanoparticle Structures”. In: *Journal of Chemical Information and Modeling* 63.2 (Jan. 2023), pp. 459–473. ISSN: 1549960X. DOI: 10.1021/acs.jcim.2c01203.

[72] M. Methfessel and A. T. Paxton. “High-precision sampling for Brillouin-zone integration in metals”. In: *Phys. Rev. B* 40.6 (Aug. 1989), pp. 3616–3621. DOI: 10.1103/PhysRevB.40.3616. URL: https://link.aps.org/doi/10.1103/PhysRevB.40.3616.