Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Fredrik Pettersen Sandvik

Applications of Data-Driven Equation Discovery to Synthetic and Experimental Data

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2021

NTNU
Kunnskap for en bedre verden

Fredrik Pettersen Sandvik

Applications of Data-Driven Equation Discovery to Synthetic and Experimental Data

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This thesis concludes my Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The work is based on simulated two-dimensional heat conduction data, and an experimental set-up with a heated aluminum plate and an IR camera to measure the temperature evolution. The data from these experiments has mainly been used to study data-driven equation discovery methods, and makes it possible to study the challenges of extending the methodology for synthetic data to real measurement data.

I would like to thank my supervisor Adil Rasheed for his great support and guidance. His experience and knowledge have not only helped me with the theoretical understanding but have also been a source of inspiration and motivation.

Fredrik P. Sandvik,
Trondheim, June 10, 2021

Contents

List of figures

List of tables

Nomenclature

DNN	Deep neural network
ET	Expression tree
GA	Genetic algorithms
GEP	Gene expression programming
GP	Genetic programming
IS	Insertion sequence
$LASSO$	Least absolute shrinkage and selection operator
MSE	Mean squared error
ORF	Open reading frame
PCA	Principal component analysis
PDE	Partial differential equation
$PGML$	Physics-guided machine learning
RIS	Root insertion sequence
$STRidge$	Sequential threshold ridge regression
SVD	Singular value decomposition

Abstract

Physics-based modeling can be highly accurate and interpretable, but requires accurate knowledge of the system dynamics desired to model. Data-driven methods can rely on observational data instead of knowledge, but often suffer from low interpretability and generalizability due to their typical black-box nature. By using symbolic or sparse regression for equation discovery, the system dynamics can be modeled in the form of an expression and thereby reduce some of the limitations associated with data-driven methods.

This research consists of two main elements that will be studied in parallel throughout this thesis. One of them is a two-dimensional heat conduction simulator to create synthetic data with an adjustable noise factor with the purpose of demonstrating the concept of equation discovery, how noise-sensitive these methods are, and how the impact of noise can be reduced with principal component analysis (PCA). The other part is a practical experiment consisting of an aluminum plate with a heating element and a low-cost IR camera to measure the temperature evolution in this plate. The purpose of this part is to apply the knowledge from the synthetic experiments to a more relevant industrial use case and to study the challenges of extending the methodology tested on synthetic data to real measurement data.

A two-dimensional heat equation was successfully recovered from data by using the sparse regression technique LASSO or by using the evolutionary algorithm gene expression programming (GEP) for symbolic regression. These data-driven methods are susceptible to noise, but can still be efficient for noisier data when combined with the dimensionality-reduction method PCA for denoising. When using a data-driven approach for describing heat evolution in the aluminum plate, heat conduction dynamics could not be extracted directly as only a more general expression capturing the overall cooling effect could be found, likely due to the large amount of noise from the low-cost IR camera. A hybrid approach based on an inaccurate physics-based model and using GEP to correct for the error could be used to improve performance for simulated data compared to a purely physics-based approach. No significant improvement was seen for the experimental set-up with this approach, probably due to noise.

Sammendrag

Fysikkbasert modellering kan være svært nøyaktig og oversiktlig, men krever omfattende kunnskap om dynamikken i systemet man ønsker å modellere. Datadrevne metoder kan i stedet basere seg på observasjonsdata og ikke kunnskap, men mangler ofte oversiktlighet og generaliserbarhet ettersom de ofte kan betraktes som svarte bokser med lite informasjon om hva som foregår mellom input og output. Ved å bruke symbolsk regresjon kan systemdynamikken modelleres i form av ligninger og dermed reduseres noen av disse begrensningene typisk for datadrevne metoder.

Den ene delen av denne oppgaven er en simulator for todimensjonale varmeligninger som brukes til å lage syntetisk data med varierbart støynivå. Denne brukes til å demonstrere hvordan ligninger kan finnes fra data, hvor støysensitive disse metodene er og hvordan principal component analysis (PCA) kan benyttes for å redusere støy. Den andre delen er et praktisk eksperiment som består av en aluminiumsplate med et varmeelement og et IR-kamera som måler temperaturutviklingen i denne platen. Målet med denne delen er å anvende kunnskapen fra forsøkene med syntetisk data til et faktisk eksempel med ekte måledata og se på utfordringene dette medfører.

Både LASSO og den evolusjonære algoritmen gene expression programming (GEP) klarte å finne en todimensjonal varmeligning fra kun data. Disse datadrevne metodene er støysensitive, men de fungerte også for mer støyfull data ved å benytte PCA for å filtrere bort støy ved hjelp av dimensjonalitetsreduksjon. De datadrevne metodene klarte ikke å hente ut varmeledningsdynamikken for aluminiumsplaten, men fant i stedet en mer generell ligning som fanget opp den totale kjølende effekten fra omgivelsene. En hybridmetode hvor GEP benyttes for å korrigere for en upresis fysikkbasert modell forbedret nøyaktigheten sammenlignet med å bare bruke den fysikkbaserte modellen. Ingen betydelig forbedring ble funnet for det praktiske forsøket med aluminiumsplaten ved å bruke en hybridmodell, trolig grunnet relativt store mengder støy.

Chapter 1

Introduction

Simulations of two-dimensional heat conduction equations with adjustable noise have been used to study how the data-driven equation discovery methods, LASSO and GEP, perform on noisy datasets. It was found that they can still recover heat conduction equations from noisy data when using the dimensionality reduction method principal component analysis (PCA) for denoising the data. This methodology was extended to an experimental set-up consisting of an aluminum plate with a heating element and a low-cost IR camera to measure temperatures. Heat conduction dynamics were not directly recovered with LASSO and gene expression programming (GEP), but a more general expression capturing the overall cooldown of the plate was found. This equation suggests that convection is the governing dynamics for the cooldown phase. A hybrid method based on an inaccurate physics-based model and using GEP to find a correction term for synthetic data improved performance compared to only using the inaccurate physics-based model. A hybrid approach made no significant improvement for the experimental set-up, likely due to large amounts of noise.

1.1 Motivation and background

Data-driven methods and machine learning can be very efficient for a wide variety of applications, but they are often black-box approaches that suffer from low interpretability. Data-driven methods such as deep neural networks are typically trained with large amounts of data and the model weights are altered

through millions of calculations. The result is a deep and complex model structure that does not show what is happening between input and output. One way of keeping the strengths associated with data-driven modeling, but at the same time increase interpretability, is to use symbolic regression to obtain a model on the form of an equation. Using white-box data-driven modeling is effective for understanding the actual physics behind the data, and the equation can be used for further simulation and predictions without the need of a deep and complex model.

Discovering an equation for how heat moves in a plate can be valuable as it can be used for predicting future temperature evolution and give more information about the materials solely from data. A model of the heat development can potentially be used for preventive maintenance, detecting leakages and cracks, and tell us more about the overall soundness of the system. Being able to model heat conduction and other forms of heat transfer from data could be highly valuable as it is an important process across many different disciplines and fields. Geology is one of the fields where heat transfer is of high importance. The conductive heat transfer process can be used to model subsurface temperature distribution and predict surface heat flow distribution [1] or to study slope stability [2].

1.1.1 State of the art

Several studies have focused on reducing the limited interpretability typically associated with data-driven methods by using symbolic regression to extract analytic equations describing complex dynamic systems [3][4][5]. By using the evolutionary method GEP and sequential threshold ridge regression (STRidge), the underlying physics can be modeled from data, both full analytic equations and hidden physics from source terms [3]. However, much of the research within symbolic regression are using synthetic data without noise. For practical usecases and applications, the data will however be noisy to some extent. As the methods used for data-driven equation discovery often are sensitive to noise, the step from synthetic data to noisy experimental data will be challenging, but valuable and important for further advancements in the field.

Purely physics-based and purely data-driven modeling are well researched top-

ics and have usecases in a wide range of fields. However, combining these approaches in order to exploit the strengths of both is far less researched, but some studies suggest such hybrid approaches have several advantages. Physics-guided machine learning (PGML) where simplified physics-based models have been used to augment neural networks by including calculated features at intermediate layers [6]. It was concluded that the PGML framework increased the generalizability of data-driven models and that it is especially useful when working with scarce data.

1.2 Research objectives and research questions

1.2.1 Objectives

Primary Objective: To develop data-driven equation discovery methods

Secondary Objectives:

- To evaluate the feasibility of using the data-driven equation discovery methods in extracting the heat conduction equation from two-dimensional simulated data
- To test the robustness of the data-driven equation discovery methods against noise and adapt the methods to deal with such disturbances
- To test the feasibility of using the methods with data collected from an experimental set-up
- To develop a hybrid method combining physics-based and data-driven modeling

1.2.2 Research questions

The guiding questions governing the research can be stated as:

- Can two-dimensional heat conduction equations be recovered from simulated data using data-driven equation discovery methods?
- How robust are these data-driven equation discovery methods against noise and what can be done to reduce the impact of noise?

- Can these methods be used with real measurement data from an experimental set-up?
- Can a better model be found by using a hybrid approach that combines a simpler physics-based base model and a correction term found with data-driven methods?

1.3 Outline of report

Chapter 2 introduces the necessary background theory needed for this project. This includes the theory needed to understand and implement a numerical model of how heat moves through a two-dimensional medium. Chapter 2 also covers symbolic regression, GEP and sparse regression for discovering equations and physics from data. As these methods are sensitive to noise, a theory section about PCA and how it can be used for denoising is also included. Chapter 3 describes the methods and the set-up needed to conduct experiments with both synthetic data from simulations and real measurements collected with an IR camera. Chapter 4 shows the results and the discussion of these, while chapter 5 is for the conclusions and the future work related to this project.

Chapter 2

Theory

The primary objective is to study heat evolution in a heated metal plate and to discover the equations that describe the underlying dynamics based on data. Therefore, the first section is about the two-dimensional heat equation to get a deeper understanding of the physics that will be modeled. The second part addresses the theory about symbolic regression and how GEP and sparse regression can be used for equation discovery from data. Finally, a brief section about PCA is included to explain this concept and how it can be applied for dimensionality reductions and removing noise.

2.1 Two-dimensional heat equation

The two-dimensional heat equation describes the dynamics of how heat is transferred through matter by conduction. It is called two-dimensional as it considers two spatial dimensions and can therefore be useful for modeling temperature in a thin metal plate. More accurately, it is called the *unsteady state* heat equation when studying the temperature evolution as a function of time. Heat is a type of energy that makes atoms and molecules move [7]. This energy spreads through matter as neighboring atoms and molecules also start moving due to collisions. This type of heat transfer is called conduction, but heat can also move in the forms of radiation and convection [8]. In the case of heat transfer in a metal plate with a non-uniform temperature profile, conduction effects should be significant. The two-dimensional heat equation can be written as [9]

$$\frac{\partial T}{\partial t} = D\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{2.1.1}$$

where T is the temperature and D is the thermal diffusivity. In heat analysis, this thermal diffusivity is a measurement of heat transfer rate and includes the thermal properties of the conducting material. It can be written on the following form [9]

$$D = \frac{k}{C\rho} \tag{2.1.2}$$

where k is thermal conductivity, C is specific heat capacity, and ρ is density.

This partial differential equation (PDE) has to be discretized to make it possible to use in numerical simulations for a two-dimensional grid. This discretization can be done by using finite difference methods. Applying a forward difference scheme for the first-order temporal term and a central difference scheme for the two second-order spatial terms yields

$$\frac{T_{i,j}^{k+1} - T_{i,j}^{k}}{dt} = D\left(\frac{T_{i+1,j}^{k} - 2T_{i,j}^{k} + T_{i-1,j}^{k}}{dx^2} + \frac{T_{i,j+1}^{k} - 2T_{i,j}^{k} + T_{i,j-1}^{k}}{dy^2}\right) \tag{2.1.3}$$

where superscript k is temporal index, subscripts i and j are spatial indexes in x-direction and y-direction, respectively, while dt is temporal step, and dx and dy are spatial steps.

Solving this for $T_{i,j}^{k+1}$, which is temperature for the next timestep at location (i, j), and assuming $dx = dy$, gives

$$T_{i,j}^{k+1} = T_{i,j}^{k} + \frac{Ddt}{dx^2}\left(T_{i+1,j}^{k} + T_{i-1,j}^{k} + T_{i,j+1}^{k} + T_{i,j-1}^{k} - 4T_{i,j}^{k}\right) \tag{2.1.4}$$

When dealing with this explicit method, the timestep, dt, needs to be sufficiently small to guarantee stability. This constraint can be written as

$$dt \leqslant \frac{1}{2D}\frac{(dxdy)^2}{dx^2 + dy^2} = \frac{dx^2}{4D} \tag{2.1.5}$$

My specialization project compared the advantages and drawbacks of purely physics-based modeling, data-driven modeling, and hybrid methods. The following discussion on physics-based modeling and how it can be related to the physics space figure is reused from this specialization project [10].

Discretizing equations in order to do numerical simulations on a computer is

considered physics-based modeling and can be an efficient and accurate approach for simple systems. However, there are several assumptions and simplifications in the process described for two-dimensional heat conduction, and it can therefore never be considered an exact solution. This is illustrated in Figure 2.1.1. The largest region in Figure 2.1.1 is the full physics space and considers all parts of physics, both known and unknown. This space could perfectly describe every process governed by physics. Inside this space, we have the understood physics, which is all physics we can observe and understand to this date. The modeled physics, which is everything we can describe with theory and equations, is a subspace of the understood physics. The inner circle is the space that covers numerical simulations. This space is a subspace of the modeled physics as there are constraints of actually implementing and simulating the modeled physics, such as computational capacity.

Figure 2.1.1 can also be used to explain what physics is lost when simulating two-dimensional heat conduction compared to a physical heat conduction experiment. No physics-based modeling can exploit the full physics space as this also includes unknown physics. Due to several assumptions and simplifications, we neither use the full understood physics space. The aforementioned equations assume uniform density, uniform specific heat, perfect insulation, and it also assumes we have perfect knowledge about these properties, which would not be feasible in an experimental set-up due to imperfect measurements and impurity of materials. Furthermore, it only considers two spatial dimensions, which is also a simplification compared to the physical world. We also neglect all other forms of heat transfer, such as radiation and convection. Such ideal circumstances would neither be present in the real world. The discretized model is also limited by computational capacity. The accuracy depends on the size of the step lengths, and it would only be perfectly accurate with infinitesimal steps, which is not possible. The equations above do not take any source term into account either. Modeling a heating element of a particular shape may be complicated and not very accurate as it is difficult to know how much of the effect is actually used for heating the conducting medium.

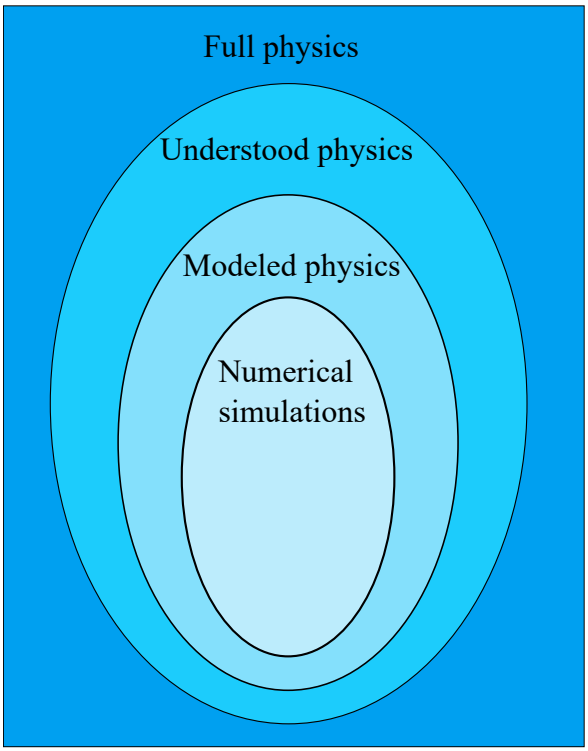


Figure 2.1.1: The actual described physics shrinks due to simplifications and assumptions

Figure reused from specialization project [10]

2.2 Symbolic regression

Symbolic regression is a class of data-driven algorithms that can find mathematical models describing and predicting hidden physics from observed input-response data [3]. While conventional regression needs assumptions on the structural form of the models and only optimize parameters, symbolic regression infers the structural form from data, as well as finding the optimal parameters. However, this approach makes the search space grow exponentially with the length of the expression. Evolutionary symbolic regression uses a preselected set of mathematical operators and operands to find the best-fit model with respect to accuracy and model complexity [3].

Unlike many other machine learning methods, symbolic regression has the advantage of being highly interpretable as it explains the relation between input

and output with mathematical expressions. This desired property is not present in black-box regression methods such as deep neural networks (DNNs). DNNs have become extremely popular due to their vast range of possible applications and high accuracy. However, as it is based on large amounts of implicit calculations and alternations on the input data through a series of hidden layers, it is not easy to extract valuable information other than the result itself [11]. The analytic solutions from a symbolic regression approach give us a greater understanding of the underlying dynamics of the system as it explains the relationship between the input parameters, both linear and non-linear dependencies. Furthermore, this can impact the generalizability as with a deeper understanding of dependencies, it is easier to adapt the models to similar applications governed by the same physics.

Symbolic regression and other data-driven modeling methods do not suffer from the same accuracy reduction due to simplifications and assumptions, as illustrated for the physics-driven approaches in Figure 2.1.1. Instead, data-driven methods use data solely so that the accuracy would be dependent on the measurement accuracy and resolution of this data, rather than restrictions and constraints due to knowledge gaps and complexity. In data-driven modeling, data from the actual system under consideration is used and can potentially account for all physics affecting the system. Figure 2.2.1 illustrates how data-driven methods collect data from the full physics space.

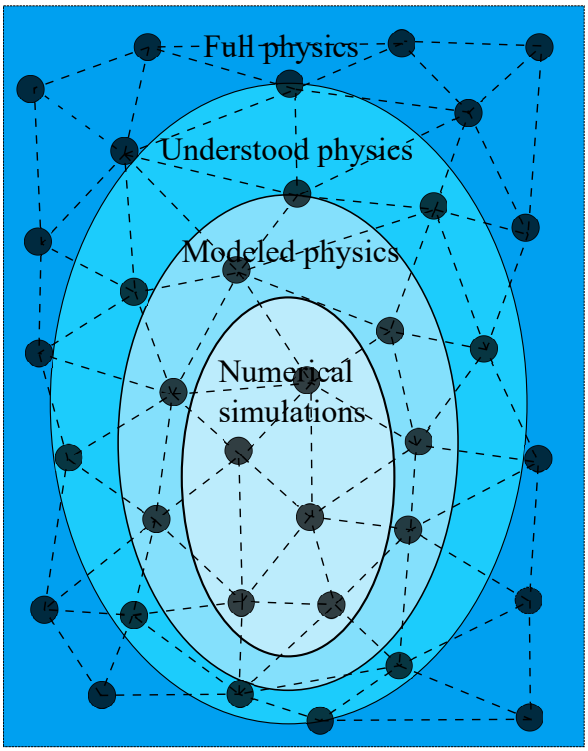


Figure 2.2.1: An illustration of data-driven modeling
Figure reused from specialization project [10].

2.2.1 Gene expression programming

Genetic algorithms (GA), genetic programming (GP), and gene expression programming (GEP) are all evolutionary algorithms that start with a randomly initialized population of individuals, selects the fittest individuals, and introduces genetic variations of these using genetic operators [12]. In GA, the individuals are linear strings of fixed length, called chromosomes, while in GP, they are nonlinear entities of different shapes and sizes, called parse trees. GEP combines these approaches as the individuals are encoded as linear strings of fixed length, which are later expressed as nonlinear entities of different sizes and shapes [12]. The flexibility and ample room for evolution make GEP a great tool within symbolic regression to find constraint-free models solely from data.

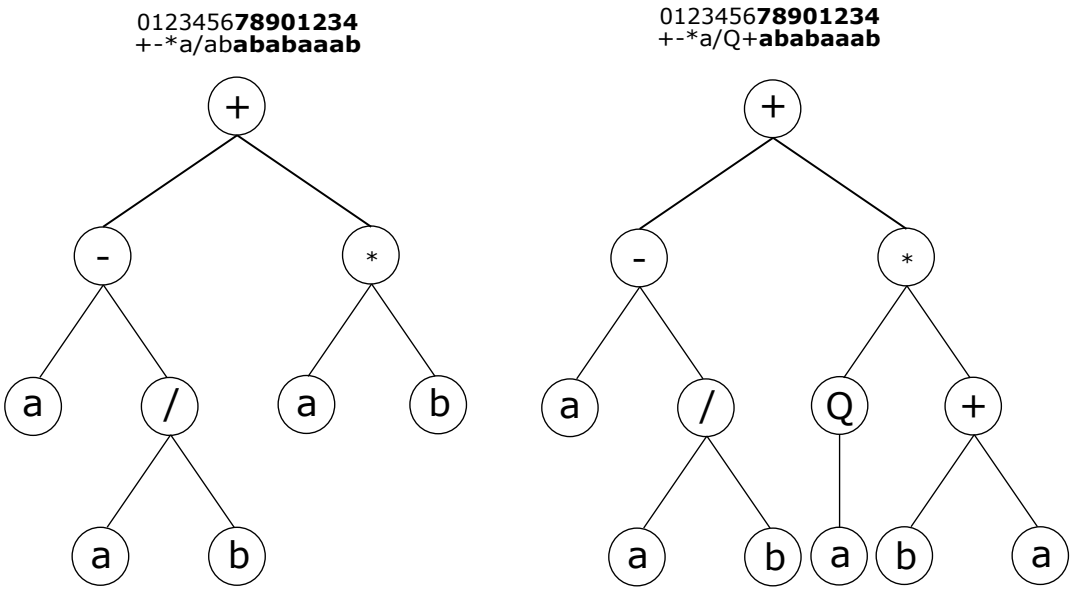
In order to build the initial chromosomes in GEP, two sets have to be defined: a function set, F, that can contain arithmetic operators, nonlinear functions and Boolean operators, and a terminal set, T, containing symbolic variables [3].

Each chromosome/gene consists of a head and a tail, where the head can be made from both the function set and the terminal set, while the tail is strictly made from the terminal set. The head length is one of the hyperparameters that needs to be tuned depending on the structure of the problem and available data. The length of the tail is then calculated as a function of the head length and the number of arguments of the function with the most arguments, n [12]. This relation is described by the following equation:

$$tail\ length = head\ length(n-1) + 1 \tag{2.2.1}$$

Several genes can be connected to form multigenic chromosomes. Typically, a more complex problem requires a higher number of genes [12].

When using GEP for symbolic regression, it is desired to evolve the chromosomes towards an expression that best fits the data. Therefore, each chromosome is expressed as expression trees (ET) and then converted to numerical expressions that can be evaluated with a fitness function to determine how they perform compared to the actual model or the measurement data. Mean squared error (MSE) is often chosen when working with regression. Even though the genes are fixed-length, the ETs can be of different sizes as they do not necessarily include all elements from the gene. Depending on the structure of functions and terminals in the gene, the last part of the tail may be a non-coding region that is not present in the ET. The coding region is often called open reading frames (ORFs). The ability to vary the ORFs and non-coding regions gives GEP high flexibility to create a wide variety of ETs even though the genes are of fixed length. This translation from a genotype structure (genes) to a phenotype representation (ETs) makes GEP a genotype-phenotype evolutionary optimization algorithm.



(2a) Chromosome A **(2b)** Chromosome B

Figure 2.2.2: Small changes to the chromosome head gives significantly different ETs

Figure 2.2.2 shows how two chromosomes can be expressed as ETs. The tail is shown in bold and should be one longer than the head, according to Eq. 2.2.1. Notice how the two chromosomes are the same length, but the ETs are not the same size. The ET from chromosome A only uses two terminals from the tail, and therefore has a nine-digit ORF and a five-digit non-coding region. Chromosome B is similar to chromosome A except that the two last terminals in the head region have been changed to two functions. A larger part of the tail is now used in the ORF, which leads to a more complex ET. The equations for the two ETs can be written as follows:

$$A : a - \frac{a}{b} + a \cdot b \tag{2.2.2}$$

$$B : a - \frac{a}{b} + \sqrt{a} \cdot (b + a) \tag{2.2.3}$$

After checking the fitness of each chromosome, a selection procedure follows to determine which chromosomes will be retained as a base for further evolution. A typical selection method is roulette wheel selection which is a proportional selection strategy that can be thought of as a roulette wheel where each chromosome has a sector, and the size of each sector is based on the fitness of the individual chromosome. As a result, a higher fitness score leads to

a higher probability of being chosen [13]. Another method is tournament selection, where a random subset of chromosomes is chosen from the population and then choosing the chromosome from the subset with the highest fitness. The selected chromosome will be copied to the next generation.

The next step is to form a new population of chromosomes by modifying the selected chromosomes from the previous generation. There are four central genetic operators to introduce variation based on the parent chromosomes [12]:
Replication: the simplest operator as the chromosome is just copied to the next generation. The fittest chromosomes are most likely to be replicated and thus be included in the new population.
Mutation: to ensure that the mutated chromosome is structurally correct, symbols in the head can be substituted with any other symbol from set T or F, while for the tails, terminals can only be substituted with other terminals in set T. Changing symbols from set T to set F or vice versa in the head leads to significant changes in the ET.
Transposition: fractions are moved to another location in the chromosome. There are three types of transposition. Insertion sequence (IS) takes a random sequence anywhere in the genome and inserts it anywhere in the head except for the first position (the root). Root insertion sequence (RIS) transposes a randomly chosen sequence from the head that starts with a function to the root of genes. In gene transposition, full genes are transposed to the start of a chromosome.
Recombination: two randomly chosen chromosomes swap some material resulting in two new chromosomes that are combinations of the parents. There are three types of recombination. In one-point recombination and two-point recombination, one and two crossover point are chosen randomly and the material downstream from the crossover point is exchanged. In gene recombination, entire genes are exchanged.

These genetic operators are tunable hyperparameters that will ensure genetic variation and evolution of the chromosomes. For example, if the mutation rate is set to 0.1, one in ten chromosomes will mutate. A chromosome can be subject to multiple genetic operators.

The whole algorithm for GEP is illustrated in Figure 2.2.3. First, a population

of random chromosomes is initialized. These are then expressed as ETs before they are executed and evaluated by their fitness. Along with the best-fit chromosome, a weighted selection procedure where the best-fit chromosomes are more likely to be chosen is used to determine which chromosomes will be the foundation for the next generation. The selected chromosomes are then exposed to multiple genetic operators to produce a new population with much variance. This cycle repeats itself until the desired amount of generations has been reached. After termination, the best-fit chromosome is saved and can be expressed as an equation.

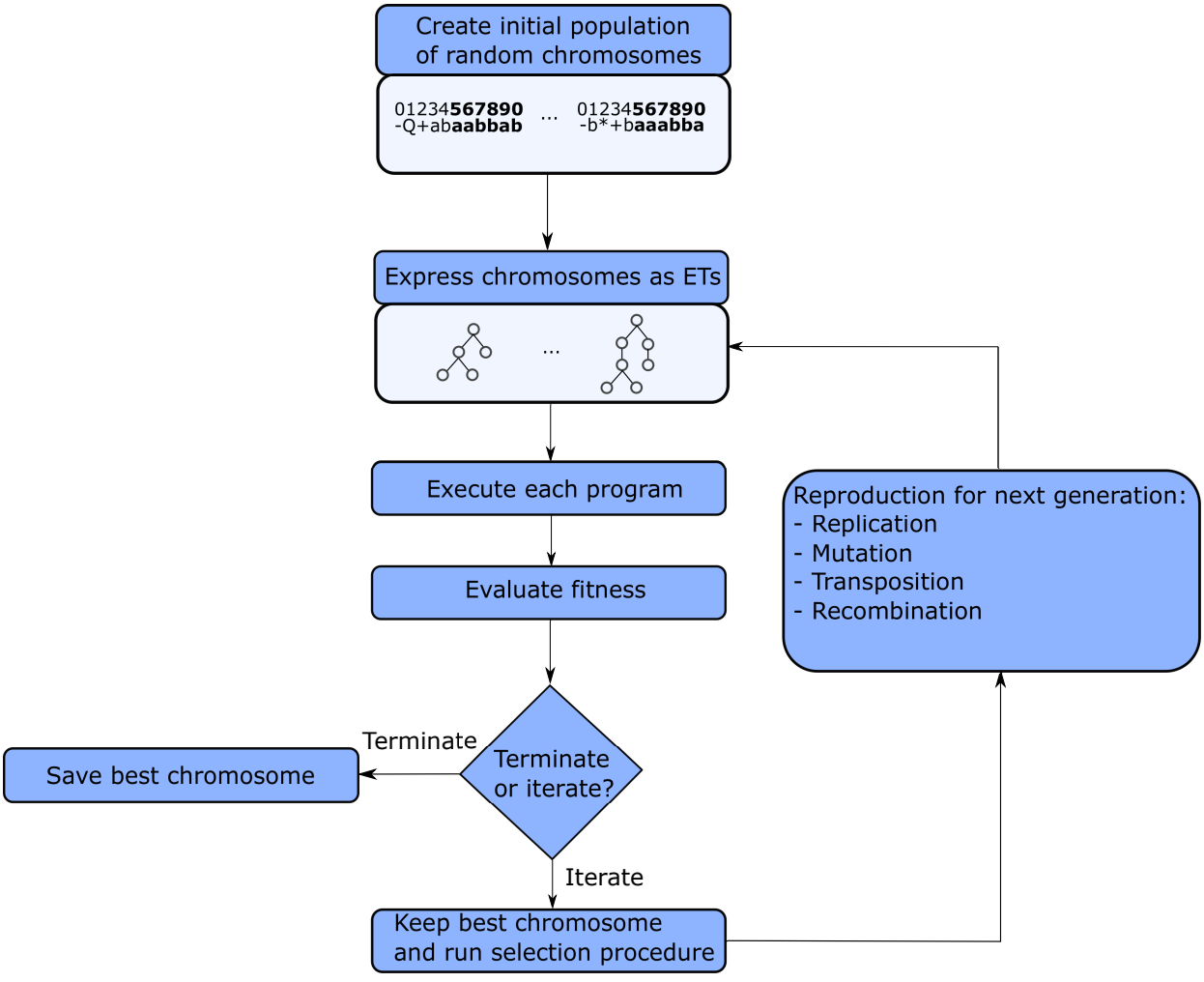
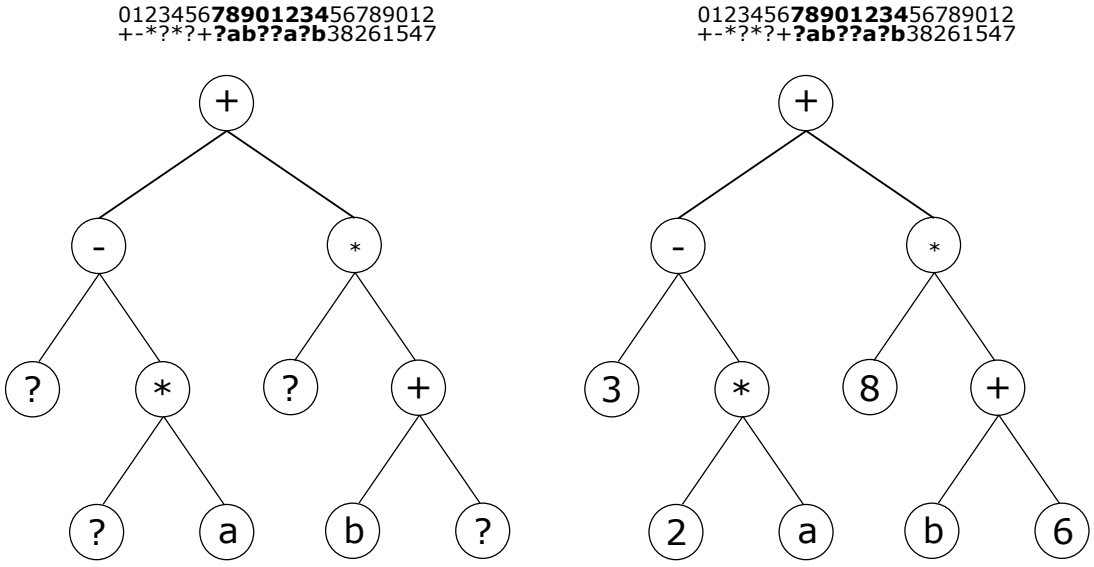


Figure 2.2.3: GEP procedure

A crucial part of symbolic regression is to find the weightings of each feature in form of a numerical constant. Ferreira addresses this challenge by introducing

a new terminal, "?", and a domain, Dc, that comes after the tail with the same length as the tail [3]. The "?" terminal represents random constants and can be used similarly to other terminals in Figure 2.2.2. These terminals can then be substituted with symbols from the Dc in a left to right and top to bottom approach as before. This process is illustrated in Figure 2.2.4 These symbols correspond to a location in an array consisting of random constants. By separating the Dc domain and the original part of the gene, the primary genetic operators are not affected by the Dc and additional Dc specific genetic operators can be introduced to ensure variation and evolution of the numerical constants. These extra operators can be Dc specific mutations, transpositions, and inversions.



(4a) The "?" terminal can be used in the ET just like the other terminals

(4b) The "?" is swapped with symbols from the Dc region

Figure 2.2.4: The process of including numerical constants in GEP

If the following array of numerical constants, C, is used, the ET from Figure 2.2.4 can be expressed as in Figure 2.2.5.

$$C = [-0.07, -0.32, 0.78, 0.45, 0.09, -0.87, 0.63, -0.42, -0.02, 0.03]$$

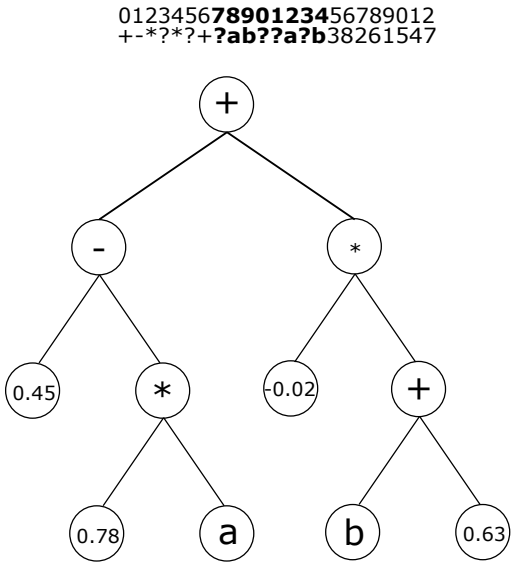


Figure 2.2.5: An ET with numerical constants

The ET in Figure 2.2.5 can be read as the following expression:

$$0.45 - 0.78a + (-0.02(b + 0.63))$$

Even though the goal of using GEP for symbolic regression is to identify both expression structure and coefficients, there is room for improvement in identifying continuous real coefficients. GEP can often find the structure quickly, but even with the described approach with numerical constant arrays and Dc specific operators, GEP still struggles with finding accurate coefficients that are not integers. A relatively simple but yet very effective improvement is to add linear scaling to the already proposed method. It has been proven that using a scaled MSE rather than the regular MSE when determining the fitness significantly improves the performance of symbolic regression [14]. The linear scaling is based on finding the optimal slope and intercept, or the weights a and b, respectively. The scaled MSE can therefore be implemented by solving

$$\min_{a,b} \frac{1}{N} \sum_{i=1}^{N} (y_i - (ay_{pi} + b))^2 \tag{2.2.4}$$

where y_i and y_{pi} are actual and predicted values for example i in the dataset.

2.2.2 Sparse regression

When discovering equations with symbolic regression, both function form and parameters are discovered at the same time. An extensive feature library is therefore required to capture the true underlying dynamics with great accuracy. However, an extensive library of features does not mean we expect complex equations to describe our model. Physical laws and equations are often relatively simple as long as they are expressed with the right features. It is desired to extract the most essential features, a process called feature selection, to drastically reduce computational demand and overfitting. This assumption of simplicity, or sparsity, can be used to recover sparse statistical models where only a few parameters (or predictors) play an important role [15].

Any PDE can be written as a linear system representation in terms of a feature library and output data [3]:

$$\boldsymbol{V}(\boldsymbol{t}) = \boldsymbol{\Theta}(\boldsymbol{U}) \cdot \boldsymbol{\beta} \tag{2.2.5}$$

where $\boldsymbol{V}(\boldsymbol{t})$ is spatiotemporal target data or measurements gathered as a single column vector. $\boldsymbol{\Theta}(\boldsymbol{U})$ is the feature library which is a matrix consisting of the features for every spatiotemporal datapoint. Each column of $\boldsymbol{\Theta}(\boldsymbol{U})$ contains all values for a particular candidate function across the full space of spatiotemporal data. $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_N]$ is the coefficient vector where β_N is the number of features. This is an overdetermined linear system as there should be a higher number of measurements than features. By finding a sparse coefficient vector, the full PDE can be represented as a weighted sum of fewer library terms. This should be a significantly simpler feature set and thus reduce the computation times and complexity significantly.

By using the least absolute shrinkage and selection operator (LASSO), the coefficient vector, $\boldsymbol{\beta}$, can be approximated with the following sparse regression objective function based on least squares: [15]

$$\beta^* = \arg\min_{\beta} \|\boldsymbol{\Theta} \cdot \boldsymbol{\beta} - \boldsymbol{V}(\boldsymbol{t})\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \tag{2.2.6}$$

where λ is a regularizing weight and $\|\boldsymbol{\beta}\|_1$ corresponds to the L_1 penalty, also called the LASSO penalty. The L_1 term is equal to the absolute sum of the coefficients. The main advantage of LASSO regression is its ability to obtain sparse

models by eliminating negligible coefficients and therefore does feature selection automatically.

Another method based on adding a penalty term is called STRidge. The goal of STRidge is to find a sparse coefficient vector that only includes active features, while the rest of the feature library is hard thresholded to zero [3]. In STRidge, the coefficient vector, $\boldsymbol{\beta}$, is approximated with the following sparse regression objective function based on least squares:

$$\beta^* = argmin_\beta \|\boldsymbol{\Theta} \cdot \boldsymbol{\beta} - \boldsymbol{V}(\boldsymbol{t})\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2 \tag{2.2.7}$$

where λ is a regularizing weight and $\|\boldsymbol{\beta}\|_2$ corresponds to the L_2 penalty, also called the ridge penalty. The L_2 penalty equals the square of the magnitude of the coefficients. These penalty terms have certain properties. An L_0 penalty is not efficient as it results in a non-convex optimization problem that is *np*-hard. Regularizing with the L_1 penalty is common for feature selection as it achieves sparsity, but relaxing the problem to a convex L_1 regularized least squares is not necessarily a good approach as it tends to perform poorly with highly correlated data [16]. L_2 regularization does not add this sparsity property, but by hard thresholding small values within a certain tolerance to zero, STRidge achieves the goal of finding a sparse coefficient vector with reduced overfitting. The values for tolerance and λ are hyperparameters that have to be tuned depending on the problem. The tolerance value can be found by starting with a very small value and gradually increasing it until the test performance decreases.

Both L_1 and L_2 can be used to regularize data as they penalize the loss function and discourage complexity of the model. Regularizing the data is important to avoid overfitting. An overfitted model will perform well on training data, but is not good at generalizing to new data. This lack of generalizability makes it unfit for modeling physics that is desired to apply to several other similar or not so similar use cases and applications.

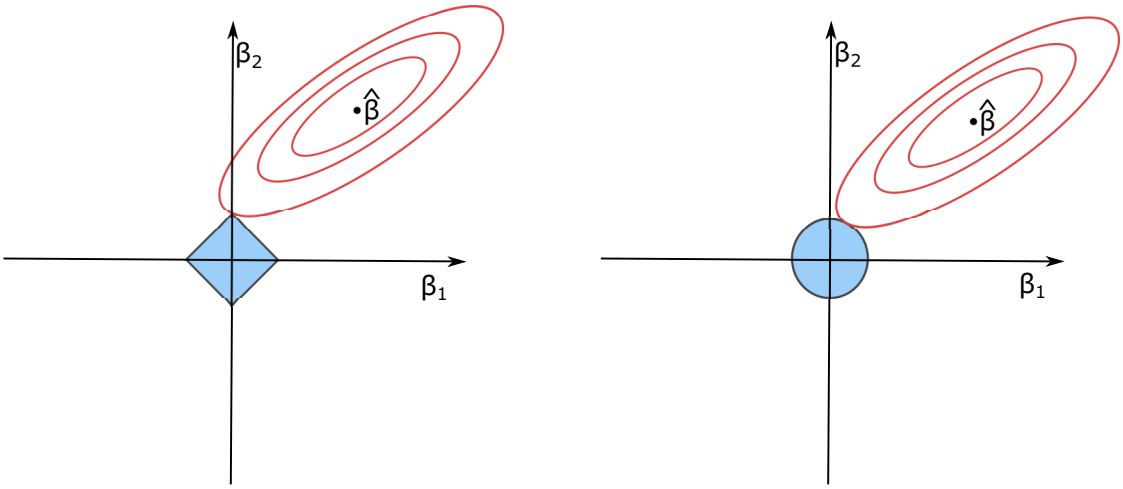


Figure 2.2.6: Comparison between LASSO (left) and ridge (right)
Figure based on Hastie et al.[15]

Figure 2.2.6 gives a visual explanation of how LASSO and ridge regression regularizes a simple two-dimensional model. The figure shows the contours of the error in red, while the blue areas are constraint functions for LASSO (left) and ridge (right). The solutions will be at the crossing points between the constraint regions and the elliptical contours. Due to the shape of the constraint function for LASSO regression, the solution often lay on one of the axes and thereby set one coefficient to absolute zero. As the ridge penalty term is quadratic, the constraint function for ridge regression will have a circular shape. This circular shape will still draw the solution towards the origin and therefore shrink the coefficients, but it will not lie on one of the axes and therefore not eliminate any coefficients. Therefore, both of these penalty terms can be useful for regularization and reducing overfitting, but only LASSO regression will lead to feature selection alone without additional hard thresholding of small coefficients to zero.

2.3 Principal component analysis

Processes and dynamics in nature can often be described with relatively simple equations, so a vast amount of information is not necessarily better — quality of information is more important. PCA is a powerful tool based on linear algebra that can reduce complex datasets to a lower number of dimensions [17]. By re-

ducing the number of dimensions, redundant information can be discarded and it is especially useful when the variables are highly correlated. This is valuable as it can remove noise and reveal simpler underlying dynamics that are difficult to extract from complex data sets. These properties make PCA an efficient tool for a variety of applications, such as image compression [18], anomaly detection [19], and face recognition systems [20].

The purpose of PCA is to project a higher-order n-dimensional vector down to a k-dimensional subspace and still retain as much valuable information as possible [21]. This yields a projection of the original vectors onto k directions, also called the principal components. After projecting the data onto the principal components, it can be projected back to the original components to obtain a filtered dataset without the contributions from the most noisy components with low quality of information. One way of finding these principal components is to find the projections that maximize variance. The direction with the highest variance is called the first principal component and it is the direction that holds the most valuable information. In other words, it is the direction that best can distinguish different data samples. The second principal component will be orthogonal to the first principal component. Equivalent to maximizing the variance, we could also minimize the projection error or the projection residuals. The first principal component, $\boldsymbol{p}_1$, can then be found by minimizing

$$\sum_{j=1}^{m}(\|\boldsymbol{x}_j\|^2 - (\boldsymbol{x}_j\boldsymbol{p}_1)^2), \qquad \boldsymbol{p}_1^T\boldsymbol{p}_1 = 1 \tag{2.3.1}$$

where $\boldsymbol{x}_j$ is datapoint j and $\boldsymbol{x}_j\boldsymbol{p}_1$ is a projection along $\boldsymbol{p}_1$. PCA for a simple two-component dataset is illustrated in Figure 2.3.1. Instead of using x_1 and x_2 to describe the data, it can be projected onto the first principal component, p_1, to explain most of the variance with only one dimension.

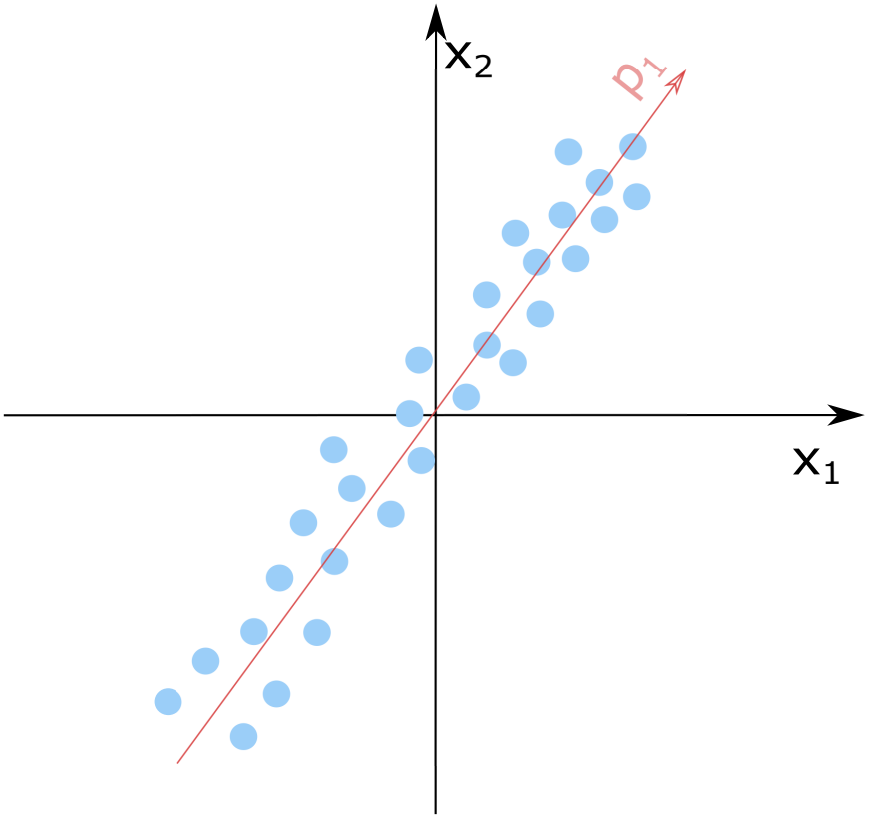


Figure 2.3.1: PCA maximizes variance or minimizes the projection error

There are several ways of deriving these principal components. It can be reduced to solving an eigenvalue/eigenvector problem or by using singular value decomposition (SVD), which is a widely used matrix factorization technique that yields a numerically stable matrix decomposition exposing useful matrix properties for extracting principal components[22][23].

Chapter 3

Set-up and method

3.1 Set-up

3.1.1 Experimental set-up

3.1.1.1 IR camera

A thermal camera allows us to do contactless and non-intrusive temperature measurements of a surface and is therefore ideal for studying the temperature evolution in a heated metal plate. Unlike regular cameras, a thermal camera detects heat, also called infrared energy, instead of visible light [24]. The thermal energy an object emits increases with the temperature. Thermal cameras usually have a lower resolution than regular cameras, as infrared energy has much larger wavelengths than visible light and therefore requires larger sensors.

The low-cost Waveshare MLX90640 110° FOV IR camera [25] was chosen for this project. This is a 32x24 IR array that gives 768 temperature measurements within its range from -40°C to 300°C. The MLX90640 was connected to a Raspberry Pi 4 Module B [26] via the I2C protocol (SDA/SCL) to access and collect these measurements. This I2C is not a hardware interface enabled on the Raspberry Pi by default, so it was enabled by uncommenting "dtparam=i2c_arm=on" in the config.txt file. To increase thermal imaging performance, the speed of the I2C communication was increased to 1 Mbit/s from the default of 400 kbit/s. This was done by adding ",i2c_arm_baudrate=1000000" on the same line. Increasing the speed above default values requires extra cau-

tion on overheating, especially if running for longer periods.

As we want the temperature as a function of time, x-coordinate and y-coordinate, $u(t, x, y)$, it is desired to gather the data as a three-dimensional array consisting of n temperature arrays of size 32x24, where n are the number of timesteps. The temperature arrays from each of these timesteps can be visualized as a heatmap by bounding the color range to a minimum and maximum temperature. According to the datasheet, it is expected that up to four sensors are defective. As this is also the case for the camera used in this project, these would have to be detected and accounted for. These defect cells either gave extremely high or low values, so they could easily be detected by checking that every value was within an expected range. This will identify both defect cells and any unexpected wrong measurements. These faulty values were fixed by interpolating the neighboring cells.

3.1.1.2 Metal plate with a heating element

The primary objective is to study how heat moves through a medium and discover the equations describing heat evolution. A thin metal plate can be used as the conducting medium to model heat transfer in two spatial dimensions. In this study, an aluminum plate that is 60 cm wide, 52 cm tall, and 10 mm thick is used. A heating element, which is a bake element typically at the bottom of ovens, was attached to the back of the plate. A purely physics-based modeling approach with numerical equations for the heating phase for this set-up would typically be very complicated due to the complex structure of the heating element. There would also be a number of assumptions that reduce the overall accuracy. Furthermore, the actual properties of the materials and heating elements might not even be known. If it instead is possible to obtain a model from data, this would be a specific model for precisely these properties and it would account for what would generally be performance-reducing assumptions.

The heating element has a maximum effect of 1200W and is usually used with a thermostat to keep a specific temperature by switching the power on and off. However, it is also desired to choose the effect of the heating element and keep it for a long period, so a simple on/off mechanism is not sufficient. Therefore, a variable transformer can be plugged into a wall outlet (230V) to adjust the out-

put to the desired voltage. This makes it possible to choose a lower effect than the maximum value that would be the case if the element was plugged directly to 230V. A thermostat is also attached to the back of the plate and a tuning knob on the side of the plate to adjust the threshold for the thermostat. One of the wires from the variable transformer to the heating element is connected via this thermometer to cut the power when the temperature is above the chosen threshold. This can be useful if we want a fixed temperature at a specific point instead of a fixed voltage for the heating element.

A problem that arises when using an IR camera for a polished aluminum plate is that the emissivity of the aluminum is actually too low for the camera to measure the temperature. Instead, the temperatures of the objects in the reflections are measured. The emissivity is the ratio between an object's ability to emit energy in the form of radiation and a blackbody's ability to emit energy in the form of radiation at the same given temperature [27]. Therefore, it is crucial to have an object with a high emissivity (preferably close to 1) to get accurate temperature measurements. Polished aluminum has an emissivity of only 0.095 [27]. This emissivity can be increased with high-emissive painting or coatings, but the low-cost and non-permanent solution in this experiment is to cover the plate with opaque electrical tape that should have an emissivity of around 0.95 [28]. Even though electrical tape is usually made from plastics which are poor heat conductors, it is only 0.18 mm thick and should therefore not affect the temperature measurements significantly. The small isolation effect should also be relatively uniform as the tape is evenly covering the entire plate.

The complete set-up is shown in Figure 3.1.1. The aluminum plate is covered in black electrical tape and is placed in front of a height-adjustable camera stand where the Raspberry Pi and the IR camera are mounted. The variable transformer to adjust the voltage is shown in the background. Figure 3.1.2 shows the heating element on the backside of the aluminum plate.

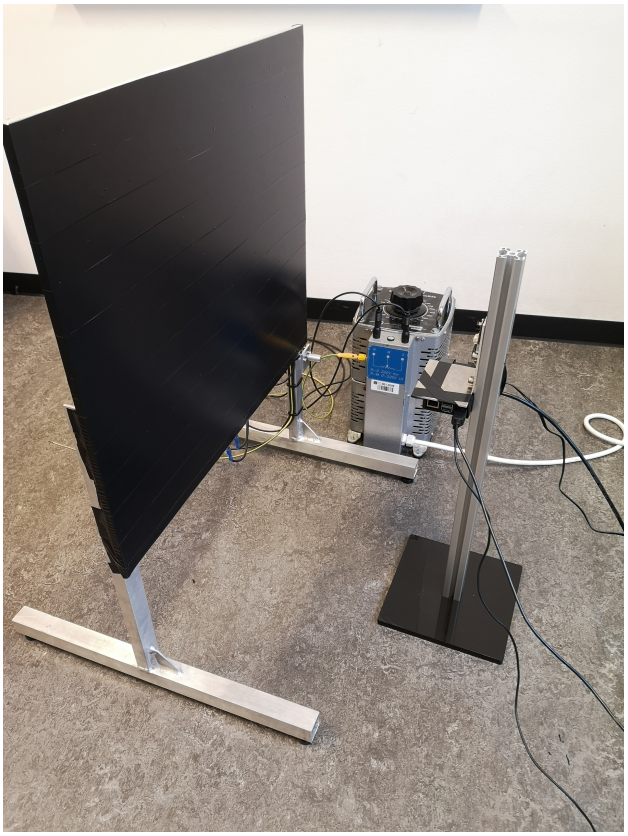


Figure 3.1.1: The full experimental set-up

Figure 3.1.2: Heating element on the backside of the plate

3.1.1.3 Experiments for data generation

When collecting measurements from the experimental set-up, the voltage of the heating element was set to 135V, which gives around 700W, and was heating the plate for one hour and 15 minutes. The plate had then reached equilibrium and the heating element was turned off. The IR camera was collecting data with intervals of 0.5078 seconds, and 9000 frames of measurements were collected for the heating phase and 9000 frames for the cooldown phase. Figure 3.1.3 shows the temperatures right after the heating element was turned off and the cooldown phase had started. The highest temperatures are slightly above 80°C. The heating element could not be used at full power as it would lead to temperatures too high for the electrical tape.

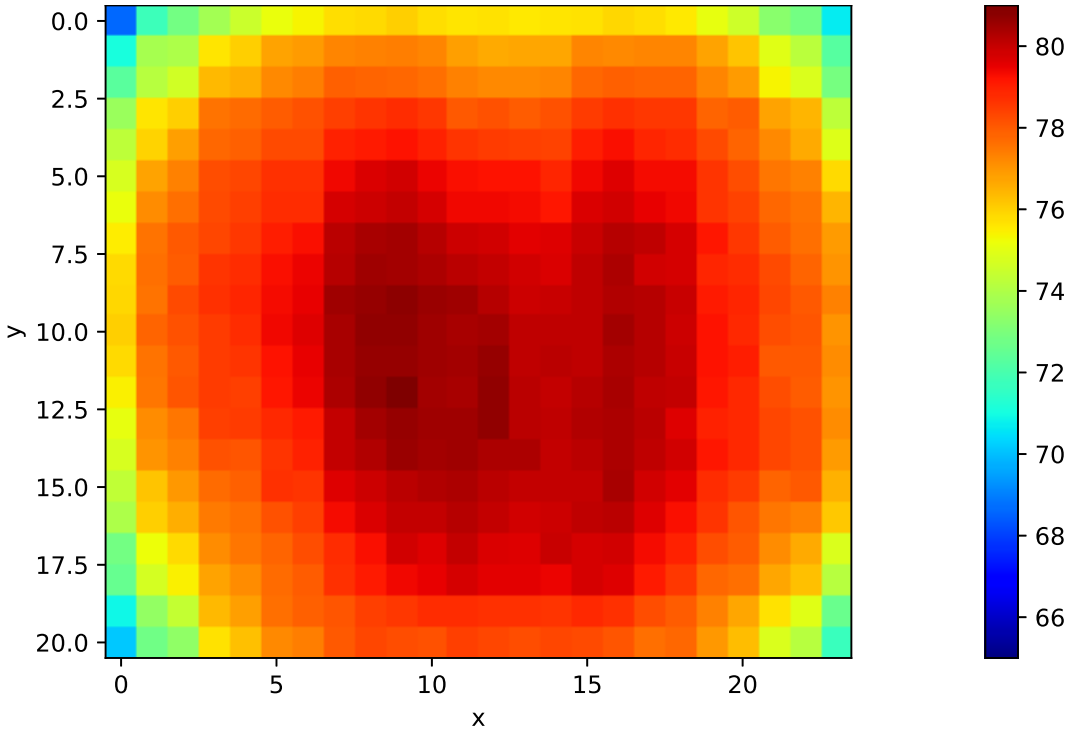


Figure 3.1.3: The initial temperatures for the cooldown phase

3.1.2 Synthetic data from simulated heat conduction

As this is a low-cost IR camera subject to relatively large amounts of noise, it is also desired to create synthetic data with the possibility of adjusting the noise. Simulated data can be used to verify that the methodology works and to

study how much noise it can handle by gradually adding noise up to the point of failure. A simulator was implemented in Python based on the theory from Section 2.1 and using Eq. 2.1.4 to calculate the next timestep. In addition, a random number within a desired range can be added here to symbolize noise. The temperature is simulated as a function of time and two spatial dimensions, $u(t, x, y)$, with the dimensions (200,50,50).

The plate is initialized with a temperature of 0°C, but with a square holding an initial temperature of 100°C to represent a heated element. The boundary conditions are set to 0°C. By calculating the future timesteps, we can study how this heat spreads throughout the plate. Figure 3.1.4 shows a heatmap of a timestep from the simulated data without noise shortly after initialization. These simulations use high values for D to obtain rapid heat conduction. This means that the heat transfer effects can be studied with relatively small datasets and computation times can be kept low when processing the data further. The other parameters were $dt = 0.1$ and $dx = dy = 1$.

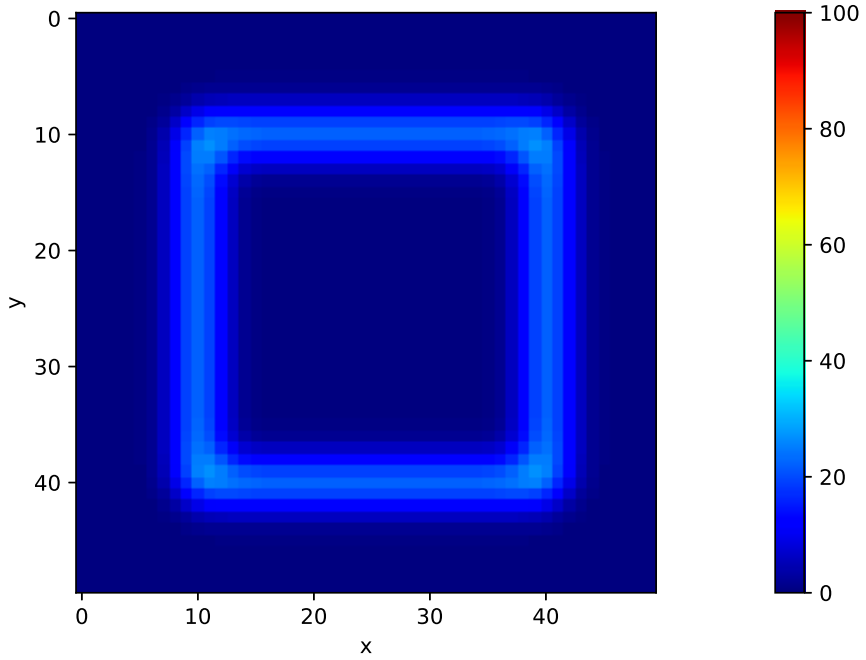


Figure 3.1.4: A timestep from simulated data

3.2 Method

3.2.1 Feature library

Both GEP and the sparse regression methods need a feature library that includes higher-order spatial and temporal derivatives to discover PDEs. The core library of features can be written as

$$\boldsymbol{V}(\boldsymbol{t}) = [\boldsymbol{U}_t] \tag{3.2.1}$$

$$\tilde{\boldsymbol{\Theta}}(\boldsymbol{U}) = [\boldsymbol{U}\ \boldsymbol{U}_x\ \boldsymbol{U}_{2x}\ \boldsymbol{U}_{3x}\ \boldsymbol{U}_y\ \boldsymbol{U}_{2y}\ \boldsymbol{U}_{3y}] \tag{3.2.2}$$

Every feature is organized as a single column with values for every spatiotemporal datapoint. The core library can be used directly as input in GEP as it is able to find combinations of the core features automatically. LASSO regression does not have this ability and it is necessary to use a more extensive feature set with every feature we can expect to be a part of the solution. This manual feature engineering step can be done by defining and calculating the following feature set:

$$\boldsymbol{\Theta}(\boldsymbol{U}) = [\boldsymbol{U}\ \boldsymbol{U}_x\ \boldsymbol{U}_{2x}\ \boldsymbol{U}_{3x}\ \boldsymbol{U}_y\ \boldsymbol{U}_{2y}\ \boldsymbol{U}_{3y}\ \boldsymbol{U}^2\ \boldsymbol{U}_x^2\ \boldsymbol{U}_{2x}^2\ \boldsymbol{U}_{3x}^2\ \boldsymbol{U}_y^2\ \boldsymbol{U}_{2y}^2\ \boldsymbol{U}_{3y}^2] \tag{3.2.3}$$

The number of features is $\beta_N = 14$, which gives 14 values for every spatiotemporal datapoint. This is still a rather simple feature set and will not find combined terms such as $\boldsymbol{U}_x\boldsymbol{U}_y$. When these methods are used on synthetic data where no such combined terms are expected, the above feature set should suffice. This need for expectations to omit potential features shows a major limitation for LASSO and ridge-based regression methods, compared to GEP, which has inbuilt feature engineering properties. For this reason, GEP will be the preferred method when working with experimental data where it is more difficult to predict what to expect.

The core features can be approximated by using numerical schemes on the collected data. A leapfrog scheme is used for temporal derivatives, while a central

difference scheme is used for spatial derivatives.

$$u_t = \frac{u_{i,j}^{k+1} - u_{i,j}^{k-1}}{2dt} \tag{3.2.4a}$$

$$u_{2t} = \frac{u_{i,j}^{k+1} - 2u_{i,j}^{k} + u_{i,j}^{k-1}}{dt^2} \tag{3.2.4b}$$

$$u_x = \frac{u_{i+1,j}^{k} - u_{i-1,j}^{k}}{2dx} \tag{3.2.4c}$$

$$u_{2x} = \frac{u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}}{dx^2} \tag{3.2.4d}$$

$$u_{3x} = \frac{u_{i+2,j}^{k} - 2u_{i+1,j}^{k} + 2u_{i-1,j}^{k} - u_{i-2,j}^{k}}{2dx^3} \tag{3.2.4e}$$

$$u_y = \frac{u_{i,j+1}^{k} - u_{i,j-1}^{k}}{2dy} \tag{3.2.4f}$$

$$u_{2y} = \frac{u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k}}{dy^2} \tag{3.2.4g}$$

$$u_{3y} = \frac{u_{i,j+2}^{k} - 2u_{i,j+1}^{k} + 2u_{i,j-1}^{k} - u_{i,j-2}^{k}}{2dy^3} \tag{3.2.4h}$$

where superscript k is temporal index, subscripts i and j are spatial indexes in x-direction and y-direction, respectively, while dt is temporal step, and dx and dy are spatial steps.

3.2.2 Gene expression programming

The GEP implementation was done in Python [29] by using the GEP framework *geppy* [30]. *geppy* is built on top of the evolutionary computation framework *DEAP* [31]. *DEAP* has great support for GP and *geppy* can be considered an extension from GP to GEP following the style of *DEAP*. As there are a lot of tuning parameters for GEP and every run can be quite time-consuming, the rates for the genetic operators are based on the values proposed by Ferreira [12] and the values used in the *geppy* documentation [30]. These can be found in Table 3.2.1. The other tuning parameters found in Table 3.2.2 are more based on trial and error. Tournament selection was used as the selection procedure. The function set consists of addition, subtraction, multiplication and division, while the terminal set is a random numerical constant (RNC) array with 100 elements. 80% of the data was used for training and the remaining 20% was used as a test set to reduce overfitting.

Genetic operator	Rate
Mutation	0.05
Inversion	0.1
IS transposition	0.1
RIS transposition	0.1
Gene transposition	0.1
One-point recombination	0.3
Two-point recombination	0.2
Gene recombination	0.1
Dc specific mutation	0.05
Dc specific inversion	0.1
Dc specific transposition	0.1

Table 3.2.1: Rates for the genetic operators

Parameter	Value
Head length	3
Genes per chromosome	1
Length of RNC array	100
Population	200
Generations	25

Table 3.2.2: Tuning parameters for GEP

3.2.3 A hybrid modeling approach

Instead of using pure modeling methods, another approach is to combine data-driven modeling with physics-based modeling to obtain a hybrid method. This can be done by using a data-driven method to correct for the error after using a simpler physics-based model based on assumptions about the system. An error term, $S(t, x, y)$, can be included in Eq. 2.1.1:

$$\frac{\partial T}{\partial t} = D\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) + S(t, x, y) \tag{3.2.5}$$

The term S captures the error between the assumed physics-based model and the actual model, including errors in heat conduction parameters and also other

effects that are not caused by heat conduction. For the simulated data, the actual value for D is known and it is therefore possible to reconstruct the equation without any S-term, but this is not necessarily the case for real use cases, so a wrong estimate of D is used to see if a hybrid approach can correct for an inaccurate estimate. Eq. 3.2.5 was solved for S by using noise-free simulated data, and setting $D = 0.5$, instead of the actual value $D = 2$. A dataset including every value for S and every neighboring temperature measurement could then be used for GEP to find an expression for $S(k, i, j)$ as a function of $u(k, i-1, j)$, $u(k, i+1, j)$, $u(k, i, j-1)$ and $u(k, i, j+1)$. An 80/20 training-test split is again used for the GEP to reduce overfitting.

Chapter 4

Results and discussions

As this study is comprised of several experiments with a high number of figures, the results are discussed consecutively throughout the chapter.

4.1 Accuracy of experimental set-up

According to Melexis' website [32], the camera should have a typical target object temperature accuracy of 1°C across its full measurement scale. Handling noise will be very important when transitioning from working with synthetic data from simulations to actual measurements from an experimental set-up. A more thorough study of the impact of noise is therefore conducted to better understand how the noise affects the readings of the metal plate. While keeping the plate at room temperature, 100 rapid measurements of the full plate were collected. The difference between the highest and lowest value for each cell can be plotted as a heat map as seen in Figure 4.1.1. The camera has less noisy measurements in the center of the images, while the edges and especially the corners have very noisy measurements. The average difference for all cells is 0.901°C. Even though the plate is kept at room temperature and should have a uniform temperature profile, the difference between the highest and the lowest measured temperature in the plate is almost 10°C.

Removing the top row, the two bottom rows, the three leftmost columns, and the five rightmost columns reduces the average difference to 0.760°C and the difference between the highest and lowest measured temperature in the plate to 2.18°C. Cropping out these cells with the most noise increases the overall

accuracy significantly, but as seen in Figure 4.1.1, there are still many cells with differences above 1°C. 1°C makes up a high percentage, especially at the beginning of the heating process when the plate holds a low temperature. The accuracy might also decrease for higher temperatures, but as it is difficult to keep the plate at a constant and uniform temperature, it was only tested at room temperature. Conducting this experiment gives an indication of how much noise we can expect from the physical set-up. This is used as a reference when adding more and more noise to synthetic data from simulated experiments. However, discovering the equations from a simulated heat conduction experiment with the expected noise is not sufficient as the physical set-up would also be affected by other effects, such as other forms of heat transfer and impurity of materials.

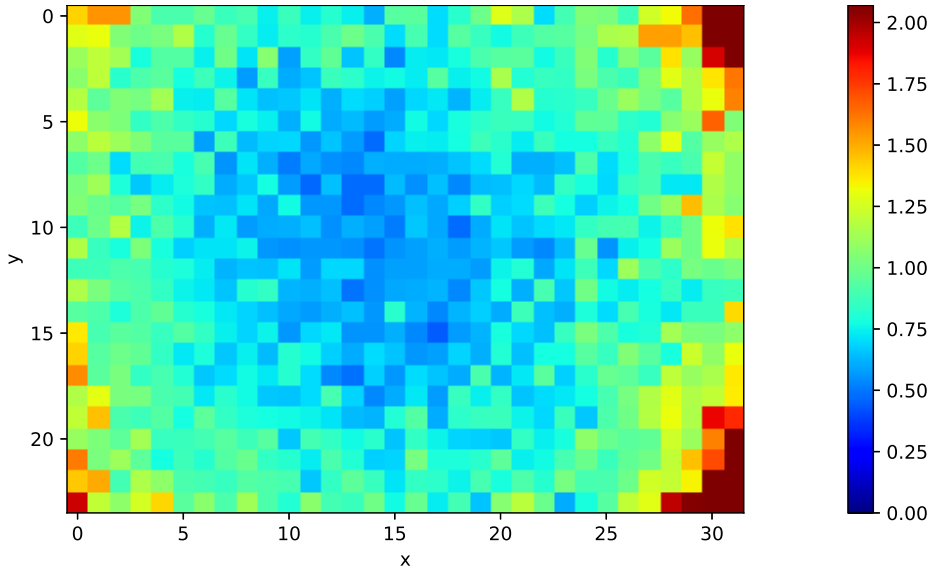


Figure 4.1.1: Accuracy of IR camera

4.2 Synthetic data without noise

A full feature set of is created with the method described in Section 3.1.2. This results in a full feature library of 15 features. As the differentiation cannot be applied to the borders, the full dataset consists of $198 * 46 * 46 * 15 = 6,284,520$ datapoints. After creating the dataset, the first time derivative, u_t, is chosen as output, y. The other 14 features make up the input, X. The goal is to find a combination of the features in X that equals y. This can be done by fitting a

linear regression model with the linear_model.LinearRegression() method from the scikit-learn package[33]. This gives the following table of coefficients:

Feature	Coefficient
u	4.63309244e-03
u_x	-1.46208262e-05
u_{2x}	2.05227318e+00
u_{3x}	-2.34106707e-06
u_y	1.58940258e-02
u_{2y}	2.03467977e+00
u_{3y}	2.86237560e-02
u^2	-2.79252754e-04
u_x^2	8.23510789e-03
u_{2x}^2	-2.01661080e-02
u_{3x}^2	-7.07926462e-03
u_y^2	7.19151780e-03
u_{2y}^2	-2.11047739e-02
u_{3y}^2	1.70244573e-03

Table 4.2.1: Feature coefficients for linear regression without noise

As seen in Table 4.2.1, feature u_{2x} and u_{2y} have the largest coefficients, while the rest are rather small in comparison. This means that the linear regression has successfully identified u_{2x} and u_{2y} as the most significant features, which agrees with Eq. 2.1.1. The actual value for D in this simulation is 2.0 and the linear regression found an expression with $2.05u_{2x}$ and $2.03u_{2y}$. Even though these have the largest coefficients, the linear regression still includes smaller values for every other feature making the full expression very complicated. The MSE between the predictions from the approximated equation and the actual measurements is 0.00126.

LASSO can be implemented by using the method linear_model.Lasso() from the scikit-learn package[33]. The alpha parameter, which is a constant that multiplies the $L1$ term and therefore affects the tolerance for setting coefficients to zero, is set to 0.001. This gives the following table of coefficients:

Feature	Coefficient
u	9.48177156e-03
u_x	0
u_{2x}	1.99258276e+00
u_{3x}	0
u_y	0
u_{2y}	1.99258299e+00
u_{3y}	0
u^2	-1.21609373e-03
u_x^2	1.03259247e-02
u_{2x}^2	-7.22007243e-03
u_{3x}^2	0
u_y^2	1.03259154e-02
u_{2y}^2	-7.22001958e-03
u_{3y}^2	0

Table 4.2.2: Feature coefficients for LASSO without noise

LASSO manages to extract the terms $1.99u_{2x}$ and $1.99u_{2y}$, which is close to the actual expression. Several of the features have been set to zero and thereby eliminated, demonstrating LASSO regression's feature selection properties. The MSE between the derived expression and the actual data is 0.00139, which is slightly higher than for the linear regression. Since the expression found with LASSO is more accurate, the low MSE for the linear regression suggests that it overfits more than the LASSO approach, which is expected.

Unlike symbolic regression, linear regression and LASSO regression can not combine features or find more complex terms and forms of the expression. In this case, a rather simple two-dimensional heat conduction equation is simulated and an expression with only significant coefficients for u_{2x} and u_{2y} is expected. However, with real experimental data, such a simple structural form is not expected as there are other forms of heat transfer as well. The goal is not only to identify u_{2x} and u_{2y} and their coefficients, but also to discover the unknown physics and a source term with a structural form that is not known.

GEP is an approach that does not suffer from these limitations. GEP only needs u, u_x, u_{2x}, u_{3x}, u_y, u_{2y} and u_{3y} as input features. GEP with linear scaling was able to extract this expression:

$$u_t = 2.0714u_{2x} + 2.0714u_{2y} + 0.0041 \tag{4.2.1}$$

The MSE between the estimated model and the actual data is 0.0015. In this case, linear regression, LASSO regression, and GEP managed to describe the dynamics of the data with relatively high accuracy. LASSO was closest to finding the true coefficients, but GEP found the simplest expression. Other alpha values for the LASSO regression were tested in order to set more features to zero, but it decreased the accuracy of the more important coefficients. Performing linear regression and LASSO regression was a quick process after the dataset with features had been made. GEP is a more time-consuming method that requires some testing as many parameters can be tuned. It could probably be tuned better to find more accurate coefficients, but for finding the structure of the expression it is very efficient even without extensive tuning. Linear regression and LASSO regression work because we calculate features we already know should be present in the equation. It can only find coefficients for all input features, while GEP can create combinations and new features based on a set of base features. This built-in feature engineering in GEP can be especially valuable when discovering physics with restricted a priori knowledge.

4.3 Synthetic data with noise

The previous section showed that the two-dimensional heat conduction equation could be recovered from data. However, it is also necessary to study the methods using noisy data as industrial use cases and experimental data always have noise. Random numbers between -0.2 and 0.2 can be added to the temperature calculations from Eq. 2.1.4. Using LASSO regression as before gives the coefficients seen in Figure 4.3.1

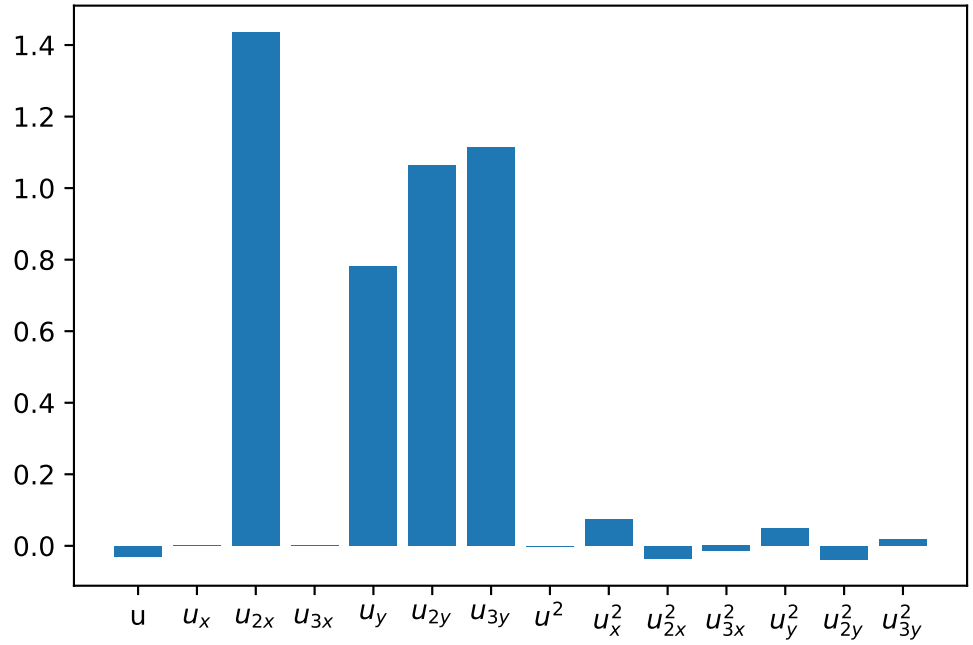


Figure 4.3.1: Feature coefficients found with LASSO after adding +-0.2 °C noise

u_{2x} and u_{2y} still have large coefficients, but they are not as dominant as they were without noise. u_{3y} actually has a larger coefficient than u_{2y}, while the coefficient for u_y is much larger than it was in the noise-free dataset. The rest of the features are less significant. It is struggling more with feature selection as fewer coefficients are set to absolute zero. The MSE is now 1.5927, which is much higher than in the previous dataset. A significant part of the dynamics is now lost in the regression.

If the noise terms are increased to +-1.0, LASSO regression with an alpha of 0.01 finds the coefficients seen in Figure 4.3.2

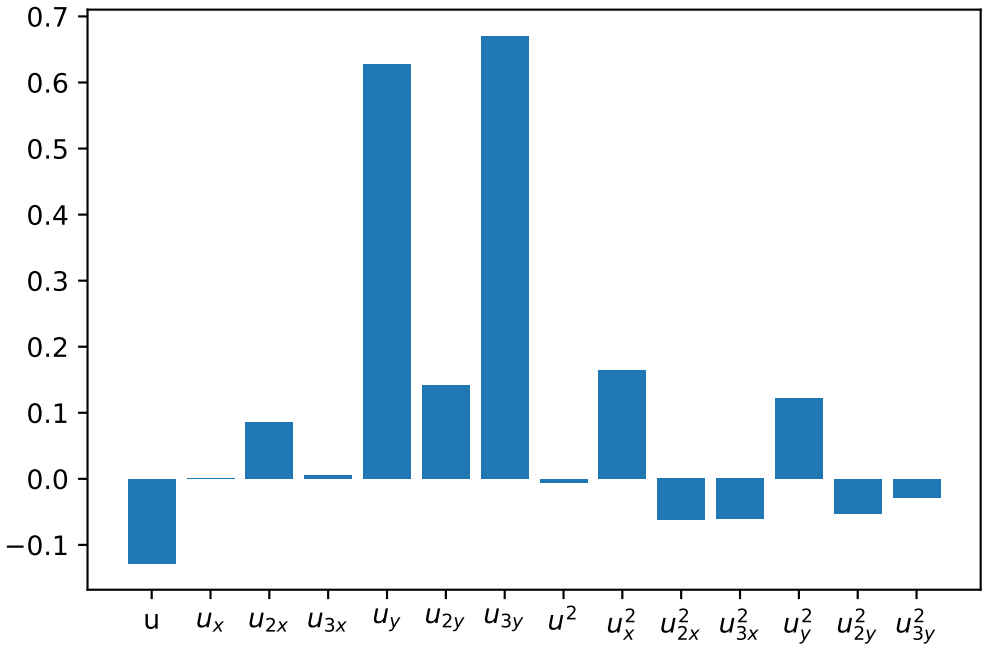


Figure 4.3.2: Feature coefficients found with LASSO after adding +-1.0 °C noise

It is no longer extracting u_{2x} and u_{2y} as significant features. Instead, it finds a complicated combination consisting of many larger coefficients. The difficulties in identifying the most important features demonstrate how important it is to keep noise to a minimum. Calculations of features and regression are both highly sensitive to noise and inaccuracies of temperature measurements can progress through the process.

4.3.1 Using PCA to handle noise

PCA can be used for denoising by discarding components that do not add valuable information as described in Section 2.3. Every timestep of the temperature data with noise between -0.2 and 0.2 was first flattened, resulting in a matrix with the dimensions (200, 2500). PCA could then be applied by using the scikit-learn package[33] for Python, which finds principal components based on SVD. Figure 4.3.3 shows how much of the variance is explained for a given number of components. There are 200 components in total but most of the variance is captured with only a few components, while the contribution from most of

them is negligible. 99.6% of the variance can be explained with only the five most important components or dimensions, shown in orange. PCA can then be used to project the data onto these five components and then back to the original components to obtain a filtered dataset.

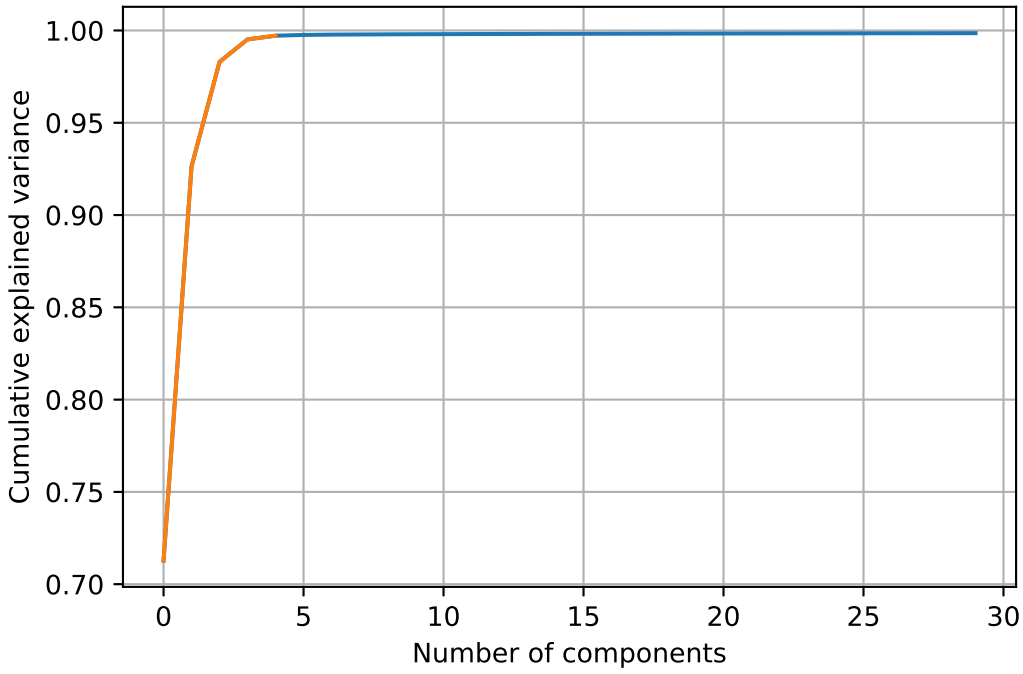


Figure 4.3.3: Explained variance for a given number of components for the dataset with +-0.2 °C noise

After filtering, the LASSO method with an alpha of 0.05 finds the coefficients as seen in Figure 4.3.4.

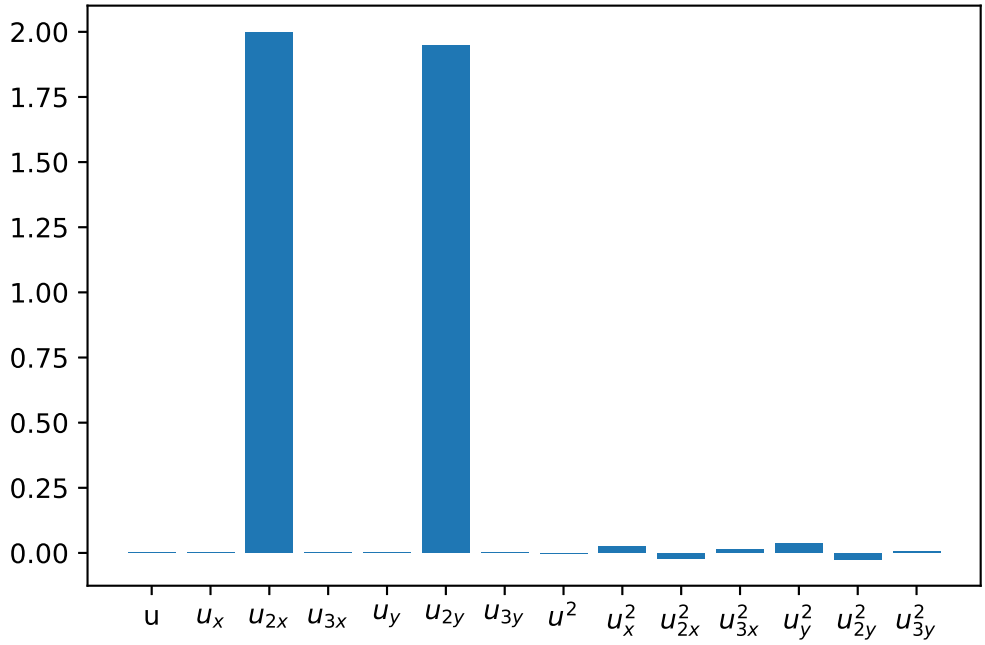


Figure 4.3.4: Feature coefficients found with LASSO after adding +-0.2 °C noise and filtering with PCA

The coefficients for u_{2x} and u_{2y} are now 2.00 and 1.95, respectively, while the rest of the coefficients are either set to zero or are very close to zero. It has therefore successfully recovered the simulated equation with high accuracy and is a significant improvement from the unfiltered data seen in Figure 4.3.1.

The same method was also applied to the dataset with noise between -1.0 and 1.0. Figure 4.3.5 shows how much of the variance is explained for a given number of components.

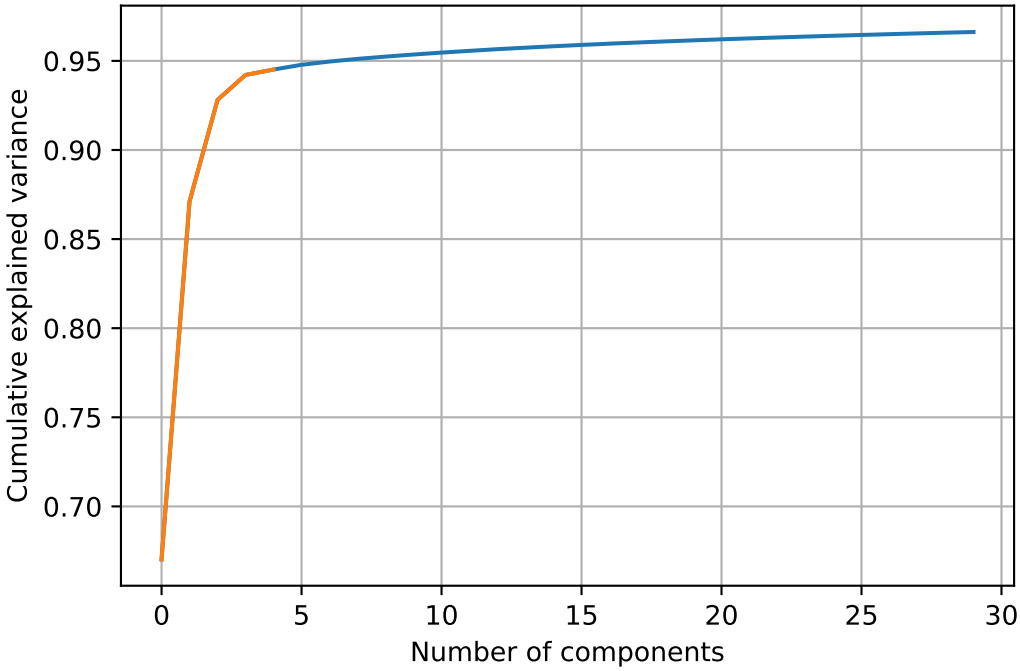


Figure 4.3.5: Explained variance for a given number of components for the dataset with +-1.0 °C noise

More of the variance is now spread over the less significant components. 94.5% of the variance can be explained with the five most important components, as shown in orange. The dataset was filtered by projecting onto these components before projecting it back. Using LASSO with an alpha of 0.05 on the filtered dataset gives the coefficients shown in Figure 4.3.6

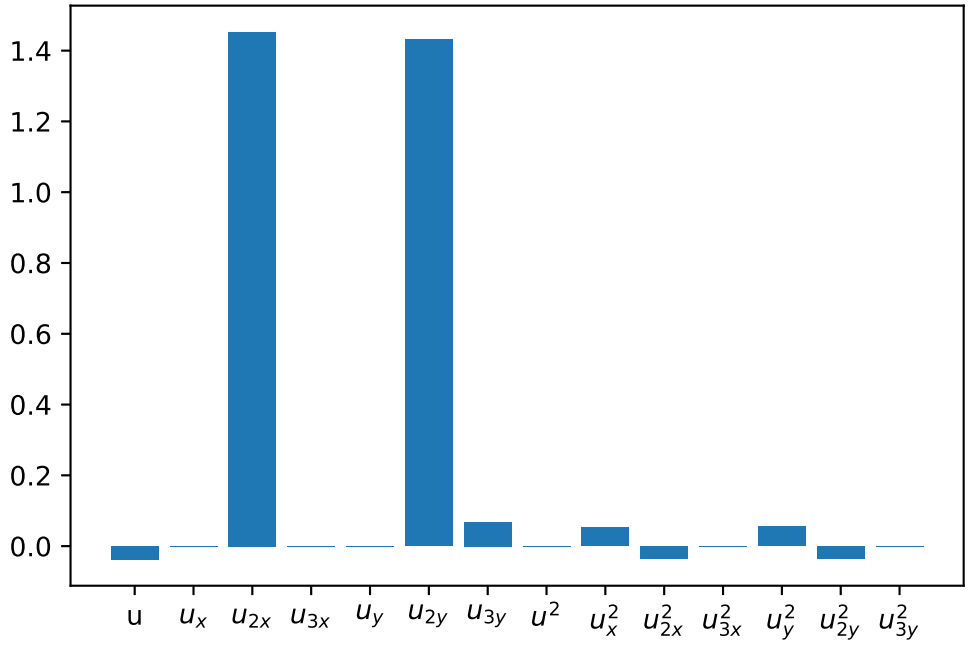


Figure 4.3.6: Feature coefficients found with LASSO after adding +-1.0 °C noise and filtering with PCA

GEP manages to find the following equation:

$$u_t = 2.1478u_{2x} + 2.1478u_{2y} + 0.0151 \tag{4.3.1}$$

It is now working a lot better than it did without PCA as LASSO identifies u_{2x} and u_{2y} as the most significant features despite the relatively large amount of noise. The coefficients for the rest of the features are zero or small in comparison. GEP finds more accurate coefficients and also a simpler structural form of the expression. These results demonstrate how effective PCA is to reduce noise and simplify datasets. The way PCA filters data is highly efficient when trying to identify and study the central underlying dynamics. Handling noise is crucial when extending the study to actual measurement data as the methods for extracting physics are very sensitive to noise. Noise-sensitive methods in combination with IR camera measurements, which are often not the most accurate data, is one of the most difficult challenges that has to be solved to make equation discovery work for practical use cases.

4.4 Equation discovery for experimental data

It has been demonstrated that it is possible to recover a two-dimensional heat conduction equation from only data, but also how sensitive these methods are to noise. Working with real measurement data from the experimental data will therefore be much more challenging. In Figure 3.1.3, which shows the initial temperatures after the cooldown phase has started, there are two warmer sections in the middle caused by the shape of the heating element as seen in Figure 3.1.2. As these measurements are taken after the plate has reached an equilibrium, there are no major differences in temperature, but it can be seen that it is slightly colder further away from the middle. Several runs with lower voltages have been collected and the following methods have also been tested for these datasets. As there were no major differences in the results, the run with 135V is in focus as it allows for the highest possible temperature variances.

The cooldown phase can be used for studying temperature evolution without additional source terms from the heating element. The most inaccurate pixels based on the accuracy study from Section 4.1 were removed, which reduces the number of data points to 8998x18x22 per feature. As heat conduction is a relatively slow process and the measurements are collected with a high sampling rate, an average of multiple timesteps can be used to reduce the noise further when finding a model. Combining 20 subsequent measurements at a time and increasing the timestep to 10.16 seconds should decrease the effect of inconsistent measurements that were observed in 4.1. After filtering it with PCA by using only six principal components, LASSO with an alpha of 0.00001 found the coefficients seen in Figure 4.4.1

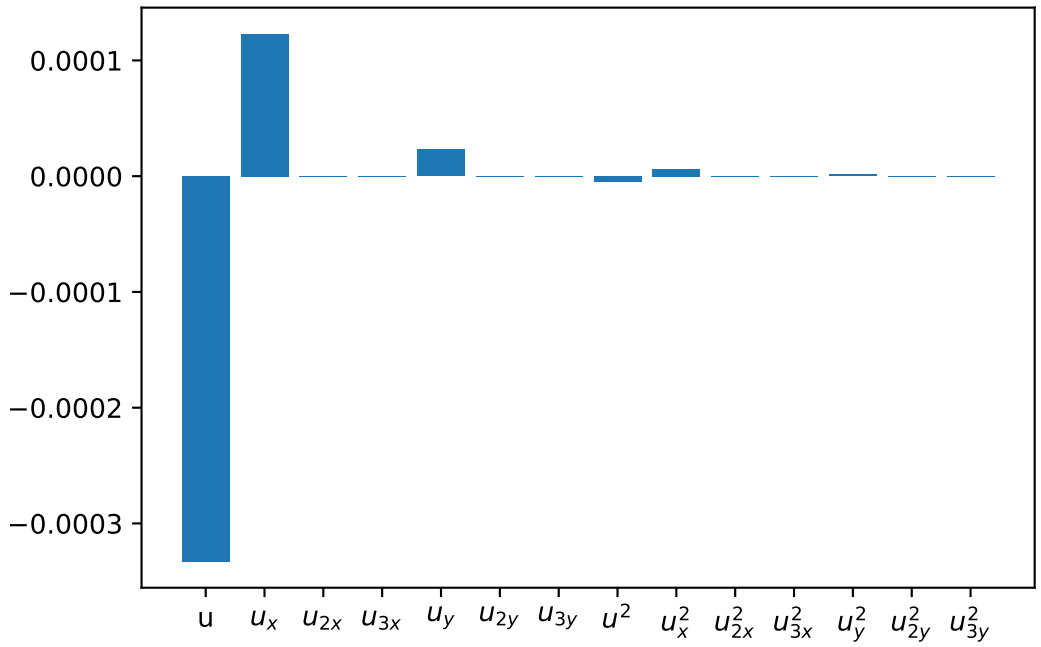


Figure 4.4.1: Feature coefficients found with LASSO on experimental data

GEP manages to find the following equation:

$$u_t = -0.00075u + 0.01991 \tag{4.4.1}$$

As the simulations used a much larger value for D than what is expected from an experimental set-up with an aluminum plate, the coefficients should be much smaller than they were with synthetic data. Unlike for the simulations, LASSO is now not extracting u_{2x} and u_{2y} as important features at all. Instead, it finds a larger negative coefficient for u, which is simply the temperature measurements in the plate. As the plate holds a higher temperature than the environment, this negative coefficient can model the cooling effect caused by radiation and convection. As these effects were not modeled in the simulations, the synthetic data did not have any heat loss but only heat transfer within the plate due to conduction. Eq. 4.4.1 is similar to Newton's law of cooling, which is an ordinary differential equation that can be written as [34]

$$u_t = k(u - u_a) \tag{4.4.2}$$

where u_a is the ambient temperature and k is a negative constant of proportionality depending on the system properties. As the ambient temperature can be assumed constant throughout the cooldown phase, and k is negative, the two equations take the exact same form. Solving for u_a yields that the ambient temperature is 26.5°C, which is likely not very far from the actual ambient temperature.

Figure 4.4.2 and Figure 4.4.3 are heatmaps from the cooldown phase at timestep 2000 and timestep 4000, respectively, where one timestep is 0.5078 seconds. As expected, it can be seen that the temperature in the plate gets more uniform after a while. The effects of heat conduction will therefore decrease significantly, and radiation and convection will be the governing forms of heat transfer, making the plate cool down more evenly. Heat conduction is not found to be very significant when trying to find an expression for the full cooldown phase.

It can be seen that the plate gets slightly warmer towards the top, which can be caused by convection. The air close to the plate heats up and rises due to the hotter air having a lower density and thereby slows down the cooling process at the top due to the higher surrounding temperatures. This is clearly a much more complex system than the simulated system making it significantly harder to find an accurate expression even with similar noise levels. GEP and LASSO find quite different expressions, and the results changed significantly even with small changes of the tuning parameters. This suggests that there might not be any simple and obvious PDE to describe the system based on the collected data. These inconsistent results might also be a sign of overfitting, even though GEP was implemented with a test set to find the best-fit equation for unseen hold-out data. The found equation does not explain the relationship between neighboring values, but is simply a decreasing function based on the previous spot measurement. It therefore discards any heat conduction effect and strengthens the hypothesis for cooling by convection to be the dominant form of dynamics.

By extracting the initial conditions and boundary conditions from the collected measurements, it is possible to make predictions by numerically calculating the temperatures in the same manner as for the simulated data. Eq. 4.4.1 was simulated numerically and Figure 4.5.2 and Figure 4.5.6 show heatmaps from this simulation. At timestep 500, it can be seen that the temperature distribution

is similar to the initial temperature profile due to the more uniform cooldown and lack of heat transfer within the plate. The lack of spatial derivatives in the prediction model is clearly seen in Figure 4.5.6 as there are large discrepancies between the boundary conditions, which are actual measurements, and the neighboring values. This lack of spatial correlation also means that only initial conditions should suffice when doing numerical simulations, and it does not need measurements for boundary throughout the full simulations. Despite the inaccurate temperature distribution in the plate, the data-driven model is still a fairly good model for the overall temperature in the plate and has a total MSE of 0.39 compared to the actual measurements.

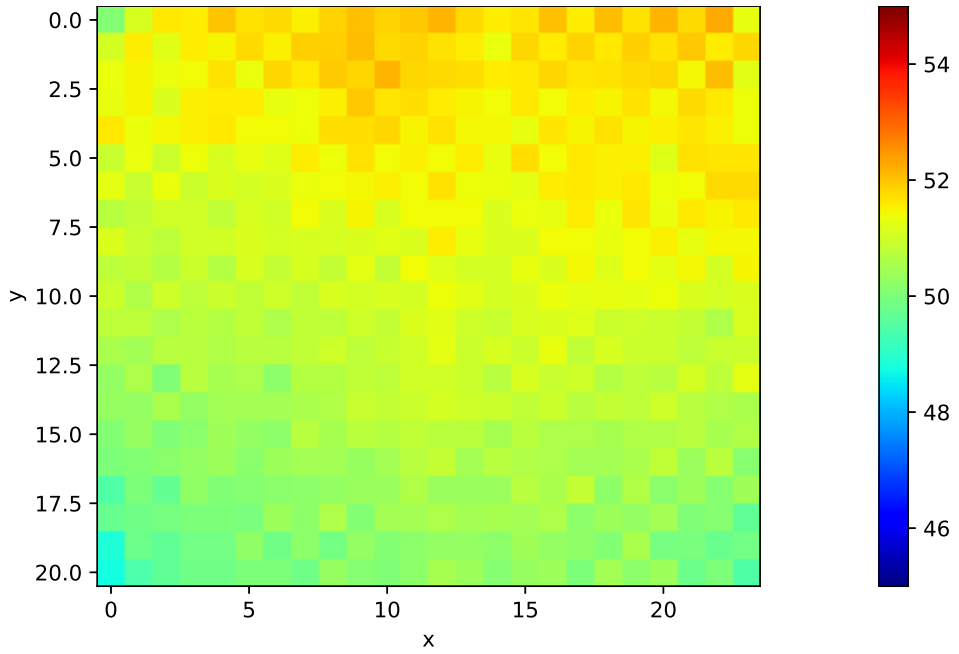


Figure 4.4.2: Temperature measurements at timestep 2000

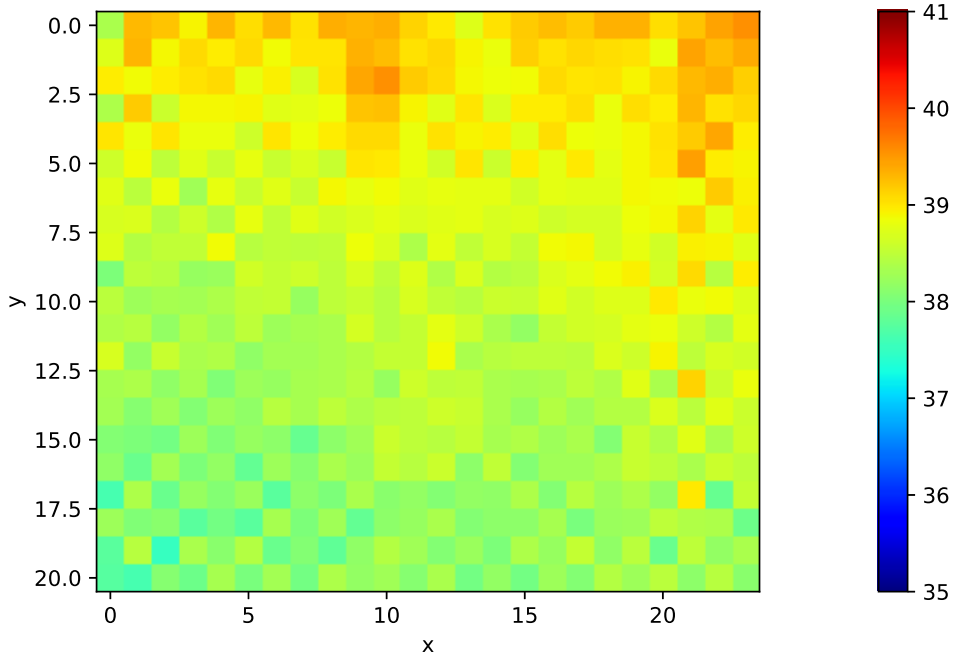


Figure 4.4.3: Temperature measurements at timestep 4000

There are several sources of uncertainty in addition to inaccurate IR camera measurements that can increase the gap between synthetic and experimental data. When calculating the features, values for dt, dx, and dy need to be known, but these are difficult to find accurately. The values used for the spatial steps, dx and dy, are estimates found by measuring the distances on the plate between the relevant pixels. Seeing this border on an IR camera requires a hotter object to distinguish it from the rest of the plate, and together with the low resolution, it becomes a challenge to find very accurate values for these important parameters. These values were measured to be $dx = 0.019$ and $dy = 0.016$. The value for dt was found by taking the total running time divided by the number of timesteps. This results in a rather accurate average value, but in reality, these values vary slightly due to some variations in the code's running time for taking measurements and the fact that the camera is on the limit for how fast it can sample. Inaccuracies of these step lengths are scaled up further when calculating higher-order features as these are calculated with exponents of these values.

4.5 Predicting with the heat conduction equation

Assuming heat conduction is the only form of heat transfer, Eq. 2.1.3 can be used with the actual physical properties of the system to calculate the temperatures at every timestep. Using $D = 9.7 * 10^{-5}$ for aluminum [35], the stability constraint from 2.1.5 gives $dt \leqslant 0.77$. The IR camera must therefore take measurements every 0.77 seconds at a minimum to guarantee stability. After overclocking the Raspberry Pi as described in Section 3.1.1.1, it was possible to sample with a timestep of 0.5078 seconds. The value for heat diffusion, D, is considered a best guess and is not necessarily accurate as the actual properties of the set-up is not known and the value also varies slightly with temperature.

The following figures show plots from actual measurements, predictions from a physics-based model using the heat conduction equation and the data-driven model found with GEP in the previous section. To study the performance of the physics-based model, plots of the difference between actual measurements and predictions are also included.

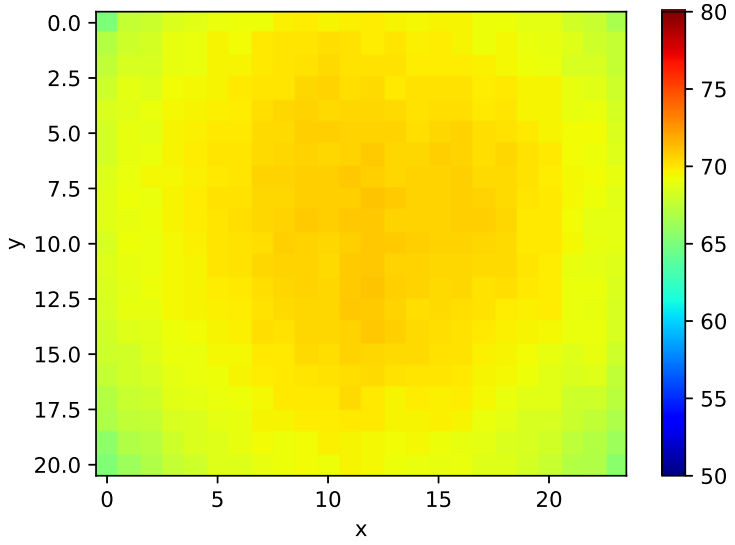


Figure 4.5.1: Temperature measurements at timestep 500

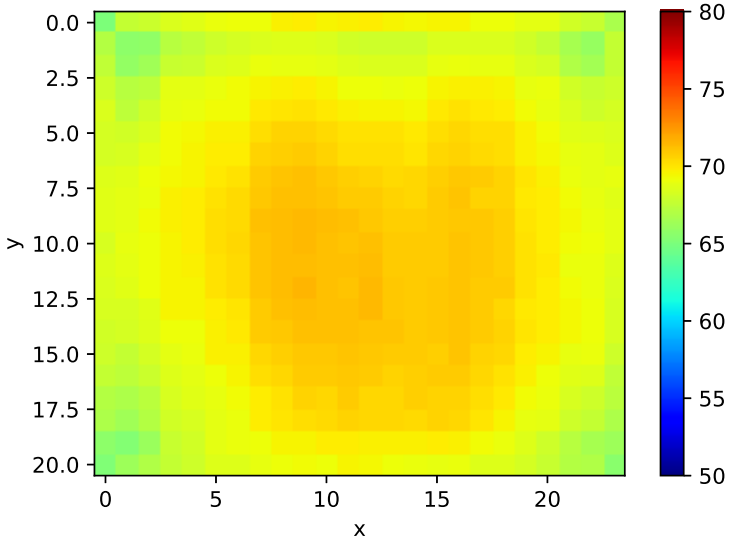


Figure 4.5.2: The prediction at timestep 500 based on the equation found with GEP

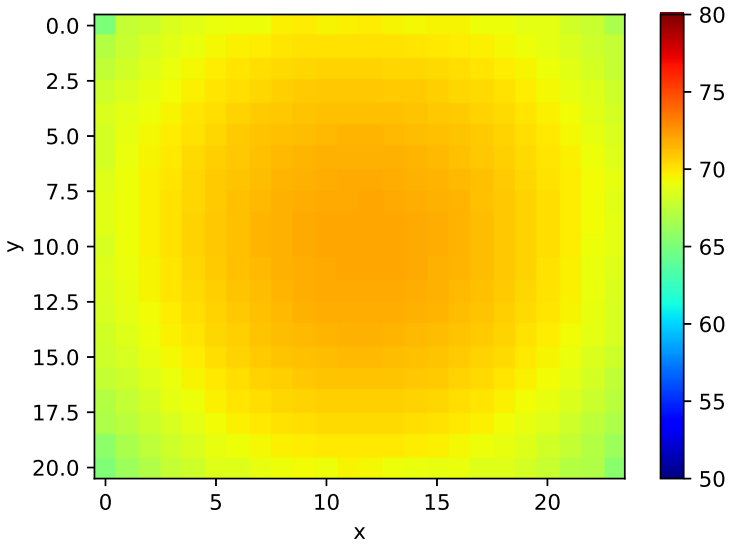


Figure 4.5.3: The prediction at timestep 500 based on the heat conduction equation

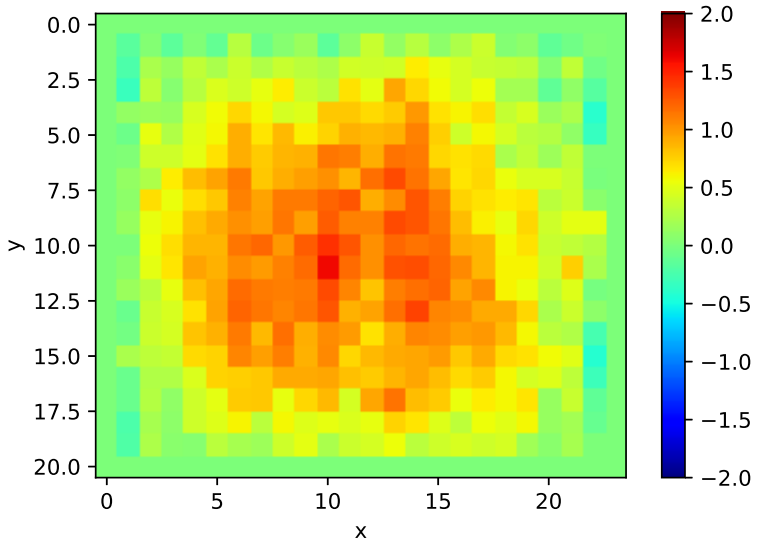


Figure 4.5.4: The difference between measurements and predictions by the physics-based model at timestep 500

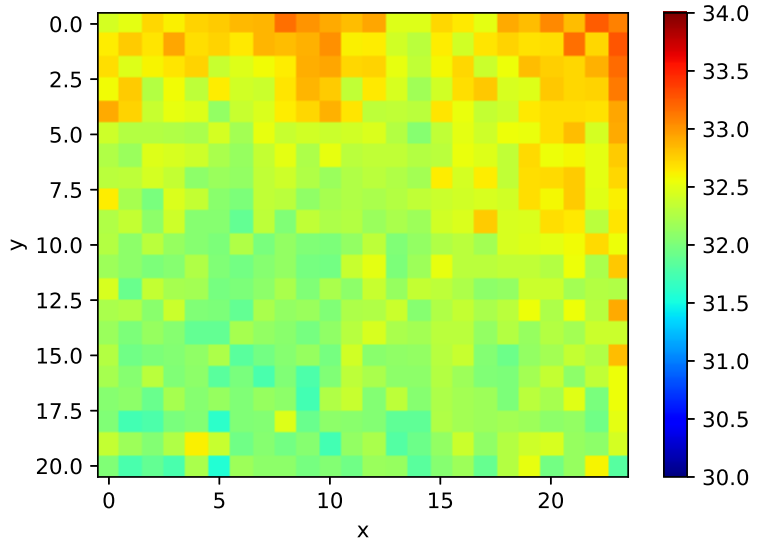


Figure 4.5.5: Temperature measurements at timestep 6000

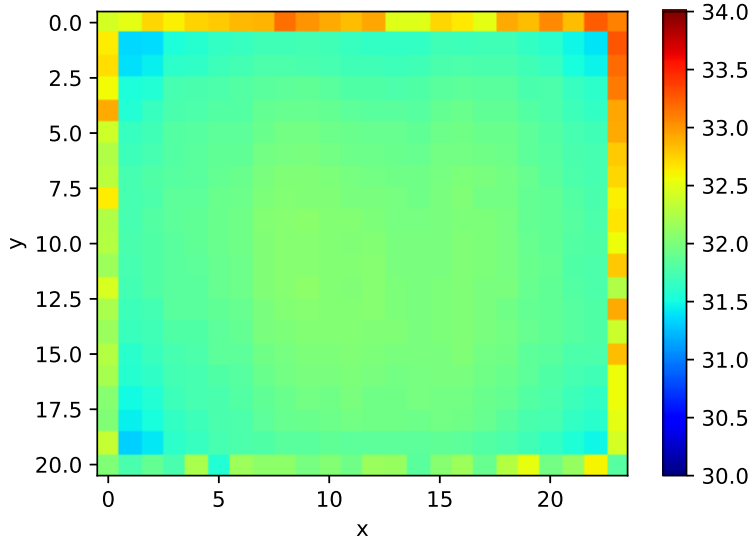


Figure 4.5.6: The prediction at timestep 6000 based on the equation found with GEP

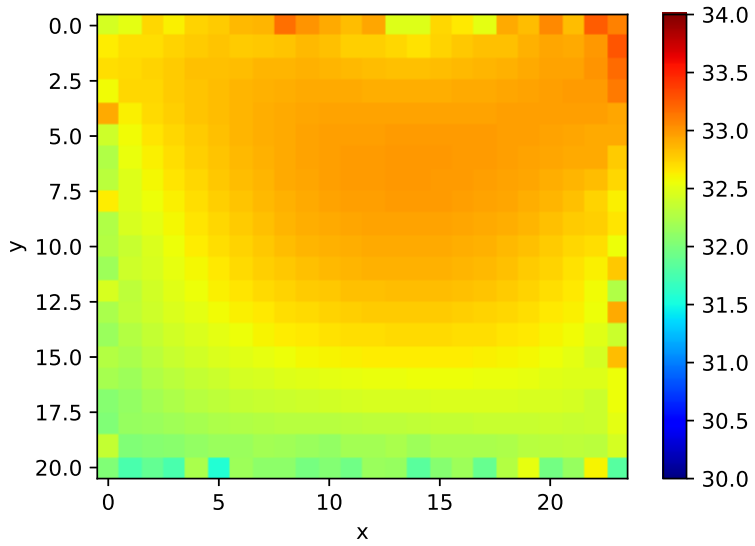


Figure 4.5.7: The prediction at timestep 6000 based on the heat conduction equation

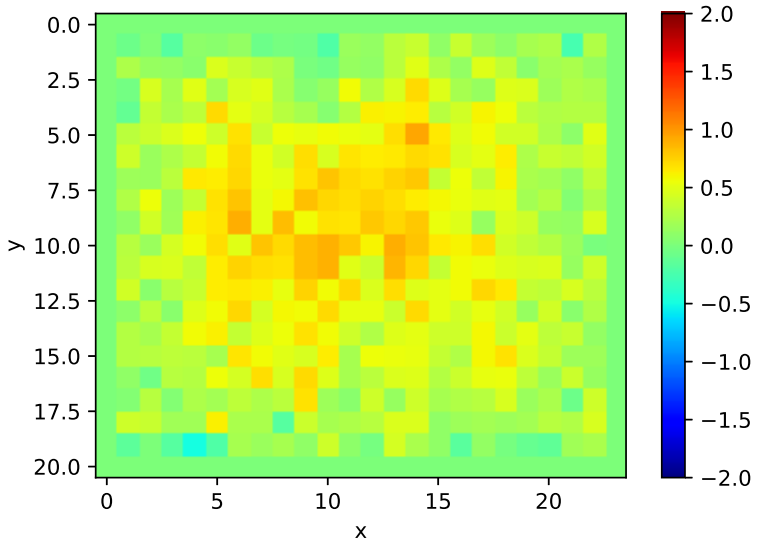


Figure 4.5.8: The difference between measurements and predictions by the physics-based model at timestep 6000

Figure 4.5.1 and Figure 4.5.3 show the actual measurements and the predictions from the physics-based model at timestep 500, which is around 250 seconds into the cooldown phase. Both figures show a warmer circular shape in the center which is reasonable due to the hot center seen from the initial temperatures in Figure 3.1.3. Figure 4.5.4 shows the difference or the error between measurements and predictions at the same timestep. The predicted temperatures are generally higher than the actual measurements, and the error is larger towards the center. This is expected as the boundary conditions are the actual measurements and the error will propagate the further away from these boundaries the model has to predict. A warmer prediction is also expected as the predictions only model heat conduction, but in reality, the plate will also be cooled by radiation and convection as the environment holds a lower temperature than the plate. These effects are captured by the boundary conditions but not in the predictions toward the center. Increasing the thermal diffusivity of the prediction model could partially substitute these effects as it would push the temperature of the full plate towards the colder boundary conditions, but it is not likely that the actual thermal diffusivity of the plate is higher than the value for pure aluminum which is modeled in Figure 4.5.3.

Figure 4.5.5 and Figure 4.5.7 are heatmaps of the actual measurements and predictions from the physics-based model at timestep 6000, which is around

3000 seconds after the cooldown phase was initiated. Heat conduction has now lead to a more uniform temperature in the plate. As the plate's temperature at this stage is closer to the room temperature, the other heat transfer effects are slower and less significant than they were at timestep 500. The differences between measurements and predictions, as seen in Figure 4.5.8, are therefore less than they were at earlier stages. Based on the accuracy study in section 4.1, noise up to 1°C is expected. Therefore, the discrepancies seen in Figure 4.5.8 seem to be more a result of noise and less systematic error compared to earlier stages when unmodeled effects had a more significant impact. The purely physics-based model has an MSE of 0.42 compared to the actual measurements, which is very similar to the performance of the data-driven model. However, the physics-based model is likely less prone to overfitting due to a more robust foundation in physics. The effect of heat rising towards the upper part of the plate can actually be observed with the physics-based model, even though no convection is modeled. Since this effect is captured in the boundary conditions, it progresses through the neighboring values with conduction. This approach requires measurements of the boundary conditions for the full simulation period, but for many use cases these boundary conditions might be known or can be predicted.

4.6 A hybrid modeling approach

It has been seen that a purely data-driven modeling approach was successful for noise-free synthetic data but struggled with even small amounts of noise and was therefore dependent on efficient denoising. Both a data-driven, and a physics-based modeling approach using the boundary conditions from measurements could be used to make predictions with reasonable overall accuracy, but they are not able to capture all the underlying dynamics and are struggling with discrepancies within the plate. The hybrid approach from Section 3.2.3 was implemented for the noise-free synthetic dataset used earlier.

GEP finds the following expression for S when using a physics-based model with inaccurate assumptions:

$$\begin{aligned} S(k,i,j) = -5.4576e-5 * (u(k,i-1,j) * u(k,i+1,j) \\ * u(k,i,j-1) * u(k,i,j+1)) + 0.1464 \end{aligned} \tag{4.6.1}$$

This new estimate of the S-term can now be subtracted from the original S-term to correct the model. Figure 4.6.1 and Figure 4.6.2 show the S-term before and after correction at timestep 30. The correction is not perfect, but it is still a significant improvement of the inaccurate physics-based model. The mean square for every S-term throughout the simulation is reduced from 7.45 to 2.68 using the correction term found with GEP.

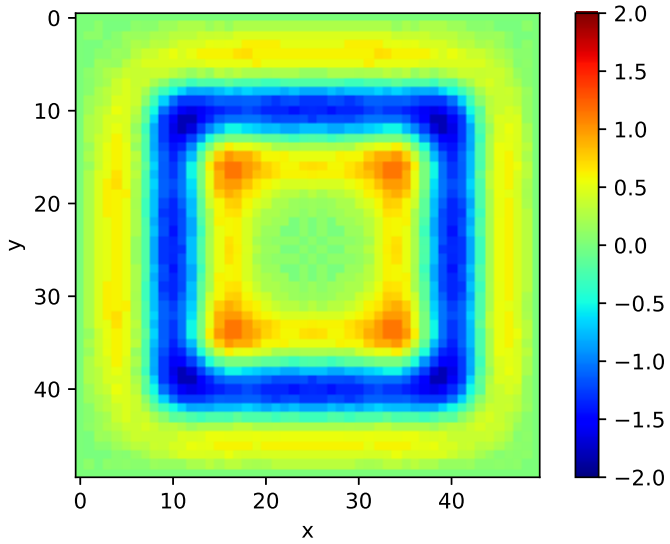


Figure 4.6.1: The error term, S, between simulated data and a physics-based model based on inaccurate assumptions at timestep 30

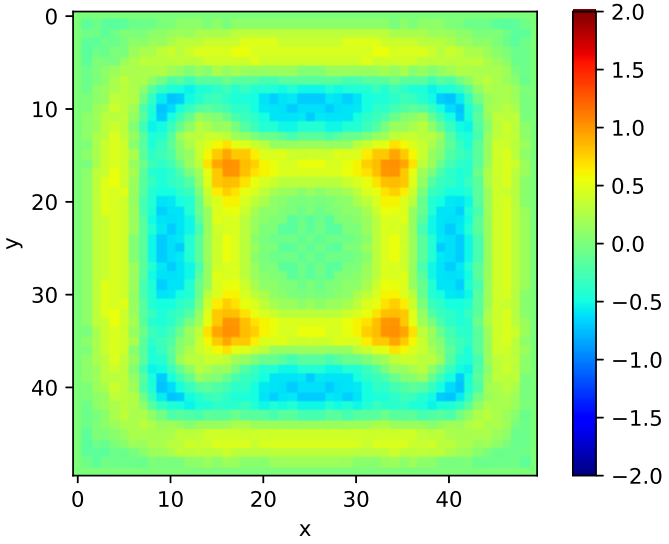


Figure 4.6.2: The error term, S, between simulated data and a physics-based model based on inaccurate assumptions after correcting with GEP at timestep 30

The improvement found with GEP can easily be used in the predictions by simulating Eq. 3.2.5 with the approximation of the error term. Figure 4.6.3 shows the simulated temperatures at timestep 30 with $D = 2$, while Figure 4.6.4 shows the temperature with a purely physics-based modeling approach based on an inaccurate assumption of the conducting material as $D = 0.5$. Figure 4.6.5 shows the temperature at the same timestep after correcting with the error term found with GEP. The simulation with a too low value for D is too hot where there were higher initial temperatures. After correction, the simulations seem to be closer to the original data. The MSE between the original simulation and the simulation with a wrong assumption of D is 14.41, while it is reduced to 7.48 after adding the correction term.

This experiment highlights a problem with pure physics-based models. If a data-driven approach is not feasible to find, a physics-based model based on assumptions is often considered the other alternative. Finding an accurate physics-based model requires accurate knowledge of the conducting material, such as thermal conductivity, k, specific heat capacity, C, and density, ρ. In addition, other physics would be have to modeled as well, which is often much more complex than only heat conduction. The model with the correction term can be considered a hybrid approach as it uses an assumed physics-based model as

a basis, but improves the performance by using a data-driven method for the residuals. In this experiment, the residual is just the error due to inaccurate parameter values for heat conduction, but in practical use cases, the S-term will also capture unmodeled physics, such as radiation and convection effects or unknown physics that is difficult to account for.

The hybrid method in this experiment did improve performance from the assumed best-guess physics-based model. It is more difficult to compare it directly to the purely data-driven methods that managed to find a near perfect recovered model from the same dataset. The hybrid approach is trying to correct an error from the physics-based model by finding a term with only neighboring temperature measurements and not a larger feature set as was used for the purely data-driven methods, resulting in a simpler equation. The S-term can easily be added to the simulations without any further calculations of new features.

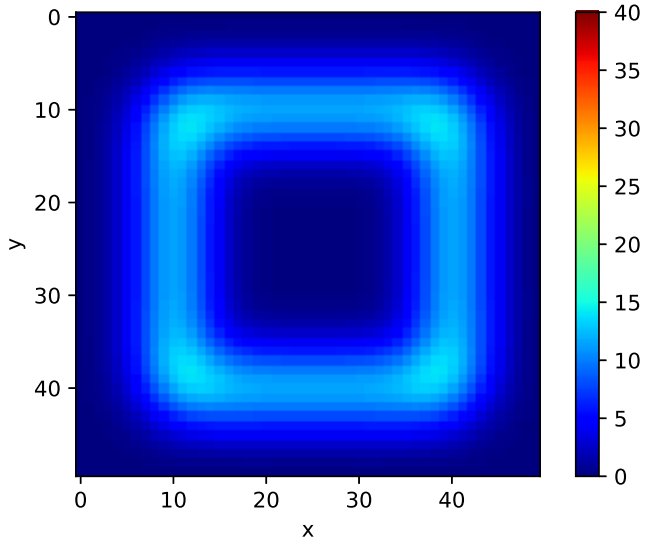


Figure 4.6.3: The original simulated temperatures at timestep 30 with $D = 2$

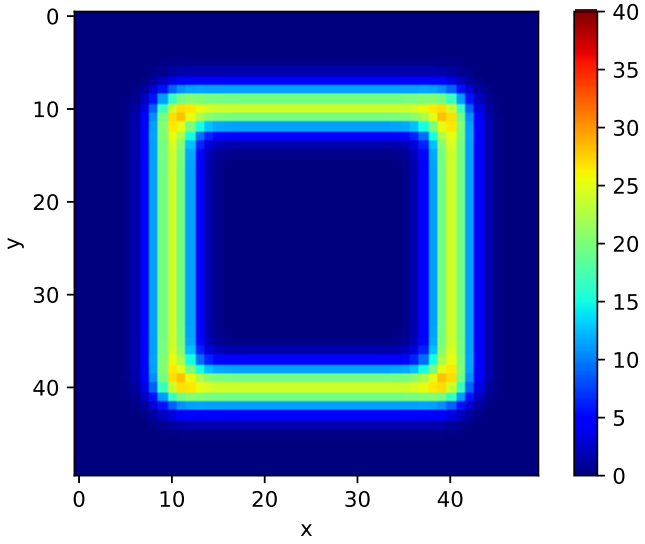


Figure 4.6.4: The simulated temperatures at timestep 30 with an inaccurate assumption of $D = 0.5$

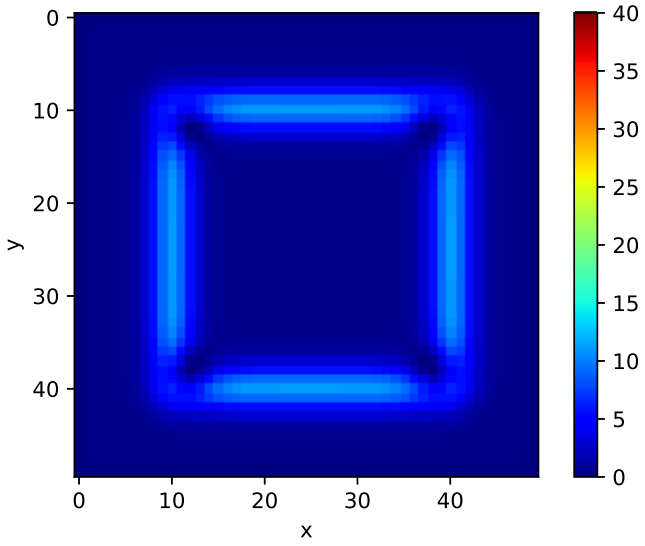


Figure 4.6.5: The simulated temperatures at timestep 30 with an inaccurate assumption of $D = 0.5$, but with a correction term found with GEP

A similar approach was tested for the experimental set-up by finding the S-term for the physics-based model from Section 4.5. Figure 4.6.6 is a heatmap of the S-term at timestep 2000 without filtering the data. There is no obvious pattern as there was for synthetic data, but it is seemingly more a result of noisy measurements. GEP had no significant progress in reducing the error through-

out the generations and found only complex equations using only one or two neighboring values with very small coefficients. These were not consistent with different parameter tunings and are most likely suffering from a high degree of overfitting. The importance of less noisy data is even higher now as most of the physics is already accounted for in the physics-based model as seen in Section 4.5. Filtering with PCA to reduce the noise in the remaining error term did neither provide any good GEP approximation of the S-term. Several of these equations found with GEP were implemented in the numerical simulations, but none of them lead to significantly improved performance. For the physics-based model to be numerically stable, the timestep needs to be sufficiently small, but the expected differences between timesteps are then minuscule. This makes the expected difference caused by heat transfer effects for this slow system almost negligible compared to the large amounts of noise.

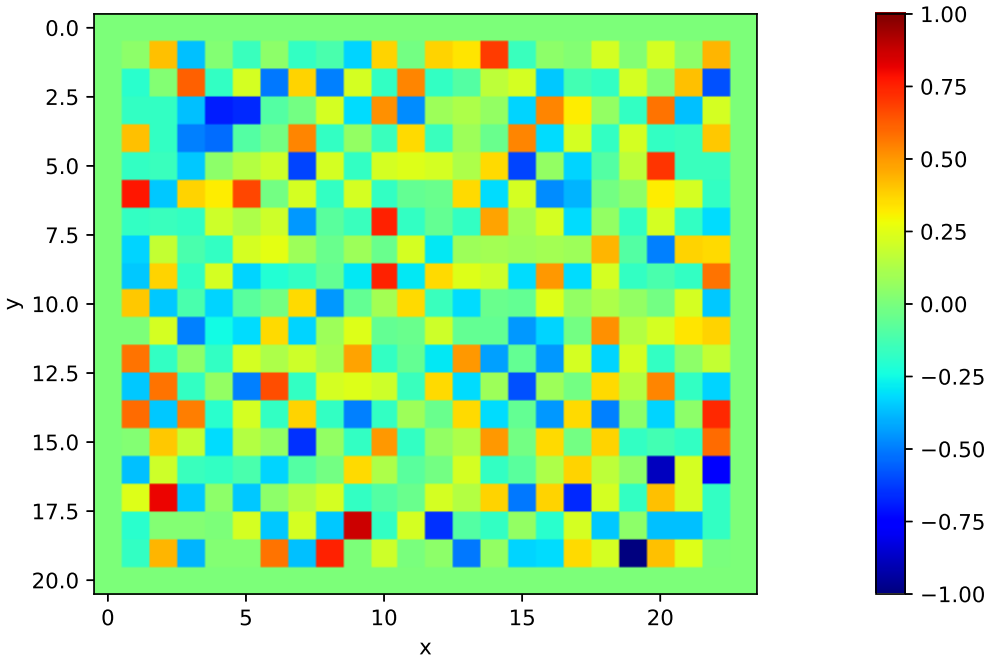


Figure 4.6.6: The error term, S, between a physics-based model and actual measurements of the plate at timestep 2000

Chapter 5

Conclusion and future work

5.1 Conclusions

The major conclusions of this thesis can be summarized as follows:

- The two-dimensional heat conduction equation was successfully recovered from simulation data by calculating a core feature set and using GEP with linear scaling for symbolic regression. It was also recovered by using LASSO with an extended feature set.
- LASSO and GEP were found to be highly susceptible to noise as they struggled to recover equations even with small amounts of noise. PCA was efficient for denoising and improved the results significantly.
- A data-driven approach using GEP found a simple equation describing the measurement data of the aluminum plate, similar to Newton's law of cooling. It was not identifying terms from the heat conduction equation, and cooling by convection seems to be the governing dynamics captured by the discovered equation. The model could be used for predictions with reasonable overall accuracy, but struggles with temperature differences within the plate due to lack of spatial derivatives in the discovered equation.
- A hybrid modeling approach where GEP was used to correct an inaccurate or incomplete physics-based model improved performance when modeling heat transfer using synthetic data, but gave no significant improvement of the noisy measurement data from the heated plate.

5.2 Future work

Part of this thesis was an attempt to find a low-cost solution for modeling heat evolution in a metal plate. It has been demonstrated that even small amounts of noise deteriorate the results but that it is possible to improve performance significantly by filtering with PCA. However, modeling all forms for heat transfer from real measurement data is much more complex than working with synthetic data made from only heat conduction. Higher complexity increases the need for less noisy data in order for the data-driven methods to extract multiple smaller physical processes. It would therefore be interesting to do a similar study with a higher-tier IR camera as this type of physics discovery can be valuable beyond a low-cost solution. Noisy data has been a major challenge with this study, and a more accurate IR camera should yield significantly better results due to the noise-sensitive nature of these data-driven methods. The challenging leap from synthetic simulation data to real measurement data becomes visible in this thesis, but most studies use only simulation data despite that most of the valuable use cases and applications are dependent on actual measurement data. Keeping noise levels to a minimum will be one of the keys to close this gap.

The experimental set-up suffers from a trade-off as a quick sampling rate is desired for the accuracy and stability of numerical simulations, but a smaller timestep makes the measurement noise much more significant as the expected temperature differences between timesteps decrease. Using an implicit numerical modeling approach can allow for larger timesteps as it it is not constrained by the same stability criteria as an explicit approach. It would also be interesting to do further studies of hybrid methods for measurement data if the noise levels can be reduced. Once a more accurate equation for the cooldown phase is extracted, this can perhaps be used for finding the source terms caused by the heating element in the heating phase.

Bibliography

[1] S. Putra, N. Suryantini, and W. Srigutomo, "Thermal modeling and heat flow density interpretation of the onshore northwest java basin, indonesia," *Geothermal Energy*, vol. 4, 2016.

[2] K. S. Myhra, S. Westermann, and B. Etzelmüller, "Modeling conductive heat flow between steep rock walls and talus slopes – thermal processes and geomorphological implications," *Frontiers in Earth Science*, vol. 7, p. 192, 2019.

[3] H. Vaddireddy, A. Rasheed, A. E. Staples, and O. San, "Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data," *Physics of Fluids*, vol. 32, p. 015113, 2020.

[4] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, "Discovering symbolic models from deep learning with inductive biases," 2020.

[5] W. Cai, A. Pacheco-Vega, M. Sen, and K. Yang, "Heat transfer correlations by symbolic regression," *International Journal of Heat and Mass Transfer*, vol. 49, pp. 4352 – 4359, 2006.

[6] S. Pawar, O. San, B. Aksoylu, A. Rasheed, and T. Kvamsdal, "Physics guided machine learning using simplified theories," *Physics of Fluids*, vol. 33, p. 011701, 2021.

[7] S. Perkins, "How heat moves." `https://www.sciencenewsforstudents.org/article/explainer-how-heat-moves`. Accessed: 2021-01-26.

[8] J. H. Lienhard, IV and J. H. Lienhard, V, *A heat transfer textbook*. Phlogiston Press, 5th ed., 2020. Version 5.10.

[9] D. W. Hahn and M. N. Özicik, *Heat Conduction, 3rd Edition*. John Wiley Sons, Inc., 2012.

[10] F. P. Sandvik, "Comparing physics-based and data-driven approaches for modeling one-dimensional heat conduction." Specialization project, 2020.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[12] C. Ferreira, "Gene expression programming: A new adaptive algorithm for solving problems," *Complex Syst*, vol. 13, 2001.

[13] J. Zhong, X. Hu, M. Gu, and J. Zhang, "Comparison of performance between different selection strategies on simple genetic algorithms," 2005.

[14] M. Keijzer, "Scaled symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 5, pp. 259–269, 2004.

[15] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*. 2015.

[16] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science advances*, vol. 3,4, 2017.

[17] J. Shlens, "A tutorial on principal component analysis," *Educational*, vol. 51, 2014.

[18] W. Perdomo and A. Méndez, *Application of Principal Component Analysis to Image Compression*. 2018.

[19] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," *Computer Communication Review*, vol. 34, 2004.

[20] B. A. Draper, K. Baek, M. S. Bartlett, and J. Beveridge, "Recognizing faces with PCA and ICA," *Computer Vision and Image Understanding*, vol. 91, pp. 115–137, 2003.

[21] I. Jolliffe and Springer-Verlag, *Principal Component Analysis*. Springer Series in Statistics, Springer, 2002.

[22] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philos Trans A Math Phys Eng Sci*, 2016.

[23] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[24] FLIR, "How do thermal cameras work?." https://www.flir.com/discover/rd-science/how-do-thermal-cameras-work/. Accessed: 2021-01-28.

[25] Melexis, "Mlx90640 datasheet." https://www.waveshare.com/w/upload/7/73/MLX90640-EN.pdf. Accessed: 2021-02-22.

[26] R. P. T. Ltd., "Raspberry pi 4 computer model b." https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf?fbclid=IwAR3TvCLunsA4-3wBlaOZaBKl-dohdFIosvBso0Q48tmz21w482wnDHGsqDU. Accessed: 2021-05-14.

[27] N. M. Ravindra, S. R. Marthi, and A. Bañobre, "Introduction to radiative properties," in *Radiative Properties of Semiconductors*, 2053-2571, pp. 1–1 to 1–8, Morgan Claypool Publishers, 2017.

[28] FLIR, "Use low-cost materials to increase target emissivity." https://www.flir.com/discover/rd-science/use-low-cost-materials-to-increase-target-emissivity/. Accessed: 2021-03-03.

[29] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009.

[30] S. Gao, "geppy: a Python framework for gene expression programming ," 2020.

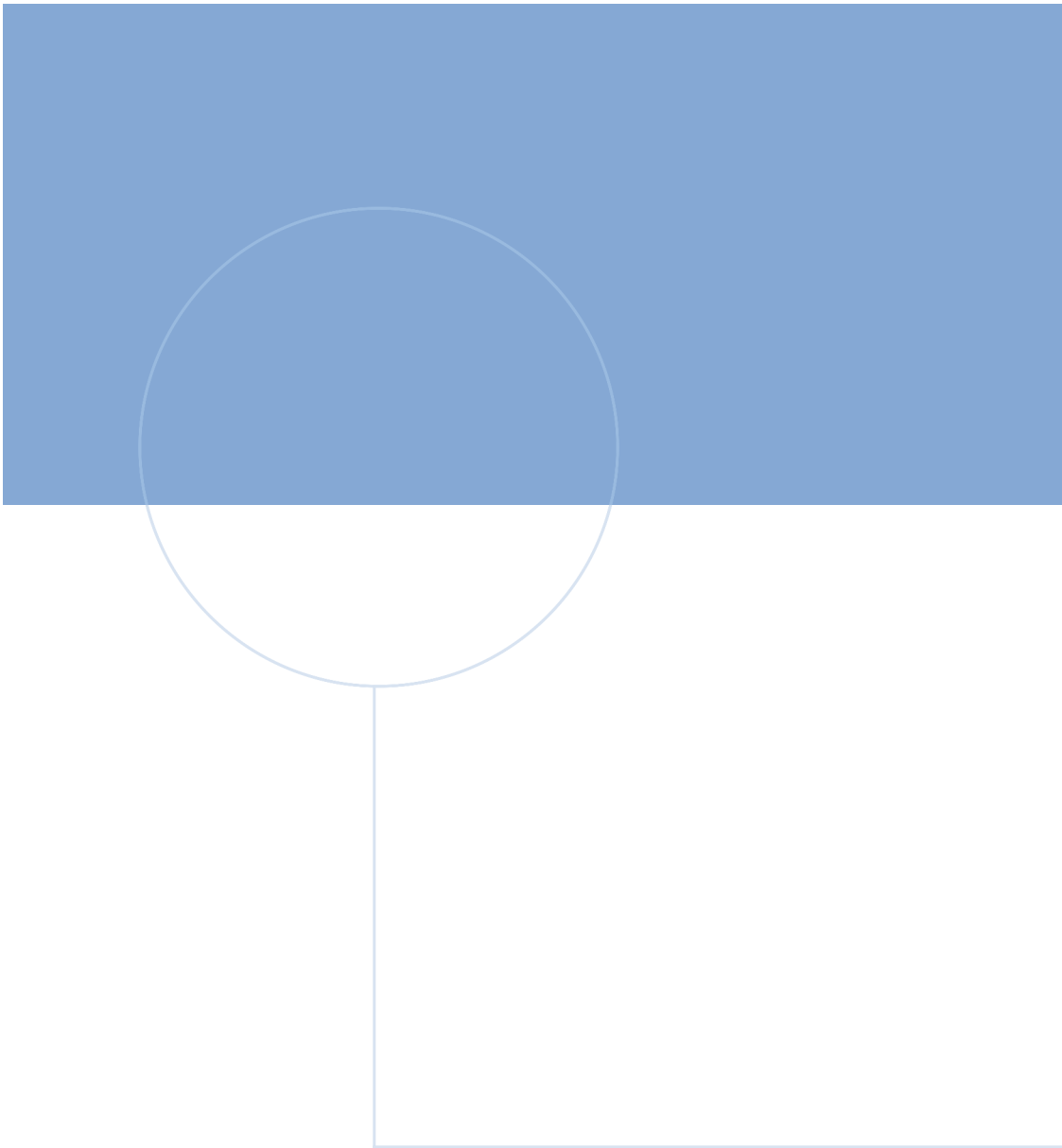
[31] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[32] Melexis, "Far infrared thermal sensor array (32x24 res)." https://www.melexis.com/en/product/mlx90640/far-infrared-thermal-sensor-array. Accessed: 2021-04-10.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] A. Mondol, R. Gupta, S. Das, and T. Dutta, "An insight into newton's cooling law using fractional calculus," *Journal of Applied Physics*, vol. 123, p. 064901, 2018.

[35] J. Nicolas, F. Martin, P. Andre, and J. Rivez, "Finite-element design of a guarded heating cylinder to measure thermal properties of materials," *Review of Scientific Instruments*, 1990.



NTNU

Kunnskap for en bedre verden