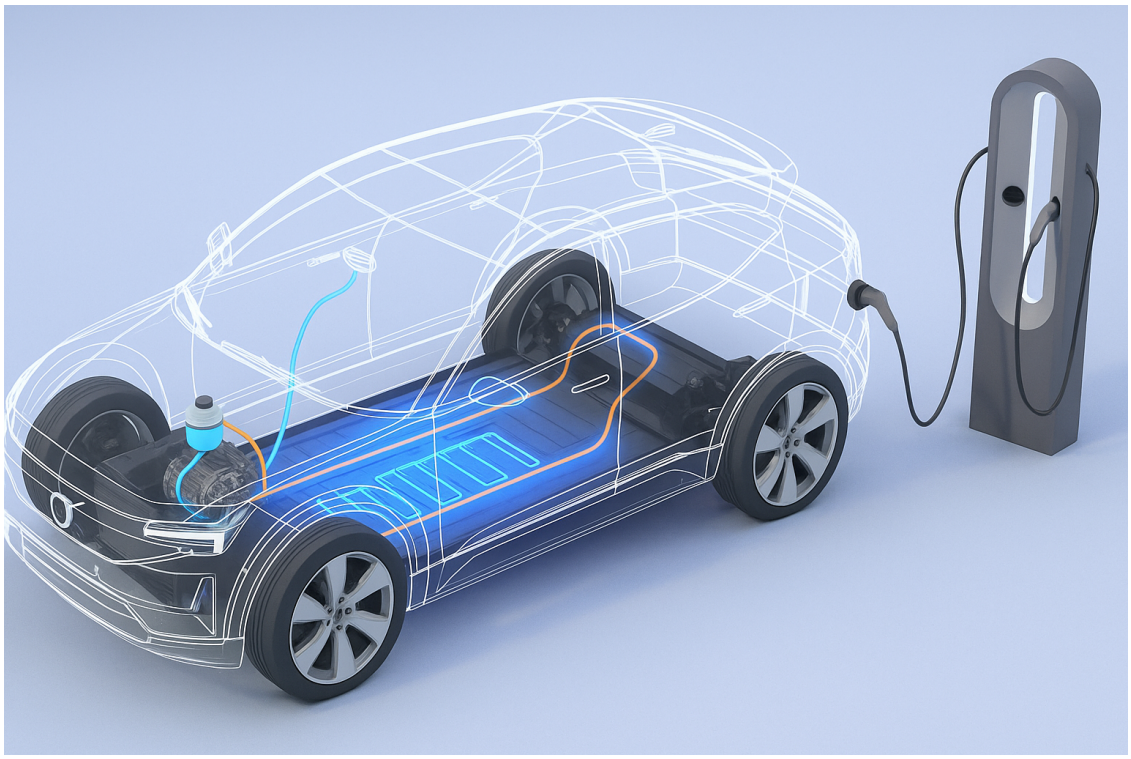
CHALMERS
UNIVERSITY OF TECHNOLOGY



# Nonlinear Model Identification for Thermal Control in BEV

## A Data-Driven Approach Using Sparse Identification of Nonlinear Dynamics

Master's thesis in Systems, Control and Mechatronics Programme

Asija Boracic

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

# Nonlinear Model Identification for Thermal Control in BEV

A Data-Driven Approach Using Sparse Identification of Nonlinear Dynamics

Asija Boracic


CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
*Division of Systems and Control*
Automatic Control Research Group
Chalmers University of Technology
Gothenburg, Sweden 2025

Nonlinear Model Identification for Thermal Control in BEV
A Data-Driven Approach Using Sparse Identification of Nonlinear Dynamics
Asija Boracic



Supervisor and Examiner: Nikolce Murgovski, Chalmers
Industry supervisors: Yashasvi Nandivada and Prajwal Prashant Shetye
Volvo Car Corporation

Master's Thesis 2025
Department of Electrical Engineering
Division of Systems and Control
Automatic Control Research Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of Volvo electric vehicle with a transparent body view, showcasing internal components of a Thermal Management System. Generated using AI (ChatGPT tools for creating image).

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Non-Linear Model Identification for Thermal Control in BEV
A Data-Driven Approach Using Sparse Identification of Nonlinear Dynamics
Asija Boracic
Department of Electrical Engineering
Chalmers University of Technology

# Abstract

This thesis investigates the use of data-driven system identification method to support control development for the thermal management system of a battery electric vehicle. The identification process is carried out using the Sparse Identification of Nonlinear Dynamics (SINDy) method combined with sequential thresholding as an optimizer. The goal is to obtain a control model suitable to use for the development of a nonlinear model predictive controller(NMPC). Several models of different complexity and accuracy are identified from recorded data and evaluated offline. To assess their ability to reach a set-point, each model is tested in a single-run optimal control problem using a direct multiple shooting approach.

Keywords:
*system identification, sparse identification of nonlinear dynamics, optimal control problem, direct multiple shooting, thermal management*

# Acknowledgements

First and foremost I want to express my gratitude to my academic supervisor, Nikolce Murgovski for his guidance and continuous support throughout this whole journey. Without his feedback and help, completing this work would have been exceptionally difficult.

I am also deeply thankful to my supervisors from the Thermal Management Department at VCC, Yashasvi and Prajwal, for their advice and support during this project. I give my thanks to my manager, Monica, for providing the opportunity to work on this topic and warmly welcoming me into the team. Additionally, I am grateful to all people from CAE Team from Volvo Cars for their advice and valuable input whenever I needed help.

Lastly, I want to thank my family and friends for their support and belief in me, not only during this thesis, but throughout my whole academic journey. Volim vas.

Asija Boracic, Gothenburg, June 2025

# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

| | |
|---|---|
| BEV | Battery Electric Vehicle |
| DMS | Direct Multiple Shooting |
| ED | Electric Drive |
| EU | European Union |
| EV | Electric Vehicle |
| HVAC | Heating, Ventilation and Air Conditioning |
| ICEV | Internal Combustion Engine Vehicle |
| IPOTP | Interior Point Optimizer |
| LASSO | Least Absolute Shrinkage and Selection Operator |
| LS | Least Squares |
| MA | Moving Average |
| MIMO | Multiple Input Multiple Output |
| MPC | Model Predictive Control |
| NLP | Nonlinear programming |
| NMPC | Nonlinear Model Predictive Control |
| OCP | Optimal Control Problem |
| ODE | Ordinary Differential Equation |
| RK4 | Fourth-order Runge-Kutta Integration Method |
| RMSE | Root Mean Square Error |
| SINDy | Sparse Identification of Nonlinear Dynamics |
| SISO | Single Input Single Output |
| SMAPE | Symmetric Mean Absolute Percentage Error |
| STLSQ | Sequential Thresholded Least Squares |
| SysID | System Identification |
| TEM | Thermal Energy Management |
| TMS | Thermal Management System |

# Nomenclature

Below is the nomenclature of indices, sets, functions, parameters, and variables that have been used throughout this thesis.

## Indices

| | |
|---|---|
| $t$ | Index for time step |
| $n$ | Number of observations |
| $i$ | Variable index |
| $m$ | Number of time samples |

## Sets

| | |
|---|---|
| $\mathcal{X}$ | Constraint set for the states of the system |
| $\mathcal{U}$ | Constraint set for the inputs of the system |

## Functions

| | |
|---|---|
| $J(\cdot)$ | Cost function |
| $g(\cdot)$ | Vector of equality constraints |
| $h(\cdot)$ | Vector of inequality constraints |

## Parameters

| | |
|---|---|
| $\lambda$ | Regularization parameter for controlling sparsity |
| $N$ | Number of control intervals |
| $T$ | Time horizon length |

| | |
|---|---|
| $k_1, ..., k_4$ | Runge-Kutta steps |
| $\mu$ | Mean of data |
| $\sigma$ | Standard deviation of data |

# Variables

| | |
|---|---|
| $\boldsymbol{x}$ | State vector |
| $\boldsymbol{u}$ | Control input vector |
| $w$ | Vector of optimization variables |
| $T_i$ | Coolant temperature measured at the $i$-th place in the system [$°C$] |
| $T_{amb}$ | Ambient temperature [$°C$] |
| $\dot{m}_i$ | Coolant mass flow measured at the $i$-th place in the system [l/min] |
| $T_{ref}$ | Reference temperature |
| $V_i$ | Valve $i$ |
| $P_{ED}$ | Speed of the pump in the electric drive circuit |
| $P_{Bat}$ | Speed of the pump in the battery circuit |
| $\xi_i$ | Sparse vector of coefficients |
| $\mathbf{X}$ | Measurements of the state matrix |
| $\dot{\mathbf{X}}$ | Measurements/numerical approximation of their state derivatives |
| $\Xi$ | Sparse coefficient matrix |
| $\Theta(\mathbf{X})$ | Library of a candidate functions matrix |

# Contents

# List of Figures

# List of Tables

# 1
# Introduction

## 1.1 Background and motivation

The development of electric vehicles(EVs) has accelerated in the past decade. The main reason is that EVs produce zero tailpipe emissions compared to internal combustion engine vehicles(ICEVs). Their widespread use in personal and public transportation contributes to reduced greenhouse gas emissions, better air quality and helps reduce dependence on fossil fuels.

The European Union(EU) has set ambitious climate goals with the objective of reaching full climate neutrality by the middle of this century. The biggest part of this strategy is the transition to electric vehicles. Besides the regulations, other factors that are contributing to the shift to EVs are concerns about energy supply and the change in customer preferences[1].

In order to meet the set targets, estimates suggest that sales of EVs in the EU need to make up about 65% of new car sales by 2030, approaching 100% by 2035. Recent data shows the progress in this transition. Electric vehicles accounted for 22.7% of new car registrations. In total, 2.4 million new electric cars were registered in 2023, up from 2 million in 2022. Registrations of new battery electric cars grew by 37%[2]. The numbers reflect an acceleration in the shift toward electrification.

Electrification is reshaping the automotive industry and simultaneously accelerating the advancement of battery technologies. As of now, lithium-ion (Li-ion) batteries are the most suitable power source for electric vehicles. Compared to alternative technologies, they are superior due to their high energy density, fast-charging capabilities and durability. On the other hand, the automotive industry is working to overcome performance barriers like battery lifetime and range, as well as technological challenges such as high cost and reliability[3].

Battery is the most expensive component in a battery electric vehicle(BEV), making up as much as 40% of the total cost. Therefore, maintaining optimal battery performance is of critical importance for ensuring both driving range and vehicle efficiency. When the battery starts to degrade, it loses capacity and reduces the lifespan of the vehicle. In this context, thermal management system(TMS) plays a crucial role as it is responsible for managing the heat generated by different components, including the battery but also power electronics, electric motor and the

passenger cabin[5]. One of the biggest challenges in the automotive industry is developing highly efficient thermal management strategies that maintain a cabin environment comfortable while keeping the battery and motor systems within their optimal temperature ranges. Such TMS not only improves the energy efficiency, but also ensures safety and extends the vehicle's driving range.

## 1.2 Related work

One paper that was analyzed is "Modeling of the Thermal Energy Management System for Battery Electric Vehicles"[6]. It presents a control-oriented model for system-level thermal energy management of the battery, heat pump system, passengers' cabin and HVAC(heating, ventilation and air conditioning). The focus was put on a cooling-down scenario for the ambient temperature $> 45°C$. The system is divided into four subcomponents: the battery thermal model, cabin thermal model, refrigerant circuit, and coolant circuit; then each component is modeled using physics-based equations. Unknown model parameters are obtained by solving an optimization problem using the least-squares method to minimize the error between GT-SUITE model outputs and the proposed model outputs. Results show high accuracy at steady-state and computational efficiency of the model. This method can be applied to similar TEM systems but there is a possibility that some re-tuning is needed to accommodate different architectures and operating conditions, so it presents a good foundation for the thesis work.

The second paper, titled "Distributed Model Predictive Controller For Thermal Energy Management System of Battery Electric Vehicles"[7] focuses on the integration of a distributed model predictive controller (DMPC) into the TEM system of BEVs. Using the ADMM algorithm, the optimization problem is solved separately for each subsystem (battery and HVAC), with local information exchanged between the subsystems to ensure optimal operation. The TEM system consists of three interconnected circuits: a battery coolant circuit, an electric drive (ED) coolant circuit, and a refrigeration circuit. The goal is to minimize energy costs related to the battery and HVAC subsystems. The results show a 2.21% reduction in energy consumption, while maintaining battery and cabin temperatures within specified bounds and minimal deviations from the desired limits.

For base knowledge in system identification, [16] and [9] offer overviews on the theory and methods, emphasizing the differences between them. For the specific method implemented in this thesis, SINDy, the key publications [18] and [19] provide detailed explanations of the algorithm, including solved examples and code implementations.

When it comes to Thermal Management, [5] gives an overview of TMS utilized in EVs, describing challenges in maintaining optimal operating temperatures for critical components in EVs: batteries, electric motors and power electronics. The paper discusses advantages and limitations of multiple cooling and heating strategies, including liquid cooling and heat pumps. Paper titled "Octovalve Thermal Management Control for Electric Vehicle"[8] presents a detailed study of the TMS of a

BEV, Tesla. Their TMS called Octovalve integrates a waste heat recovery (WHR) mechanism that can perform air conditioning and heating tasks at the same time, allowing more efficient temperature regulation. Octovalve is modeled and simulated using Simulink Simscape. The results showed reduced energy consumption with a trade-off of longer warm-up time. The paper provides an extensive analysis of the system, describing the structural setup of the valve system, heat exchange elements and fluid loops. It also addresses the operational modes of the TMS and outlines how it manages heat exchange between the cabin, battery and drive units. This is particularly valuable as such descriptions of TMS are hard to come across due to proprietary constraints.

## 1.3 Scope and limitations

This thesis aimed to develop a non-linear predictive controller for TMS with a cost function to optimize actuator energy use. The goal of the system is to quickly reach and sustain the required temperature(s) with minimal energy expenditure. Previous approaches have included both physics-based modeling in control design and the use of reinforcement learning control. The control model was needed in order to develop the controller and the idea was to build it by system identification. That turned out to be a very demanding task and it did not result in a model good enough for the control problem to be addressed within the time constraints of the thesis work, so instead of using the model for control purposes, offline assessment of models of different complexities was done and the results compared.

The work is subject to the following limitations:

- The cabin circuit (refrigerant loop) variables are not considered.
- Identification approach does not incorporate any physics-based modeling/prior domain knowledge.
- Valve actuation is idealised, assuming instant transition between positions.
- Ambient temperature $T_{amb}$ is treated as a constant over the simulation time.

## 1.4 Ethical aspects

Ethical aspects of this thesis include making sure the solution is developed in a responsible and transparent way. Any data used to improve the thermal control must be handled properly and not misused. The solution should aim to benefit a wide range of BEV users, not just owners of high-end models. If the system becomes too complex or hard to repair, it could go against the idea of sustainable and user-friendly design. Keeping these factors in mind is important to ensure the project has a positive and ethical impact.

# 2

# Theory

## 2.1 Thermal Management System in BEV

TMS in a BEV is primarily responsible for temperature regulation of various components, ensuring optimal performance, efficiency and to some extent, safety of the vehicle. Power-dense components(battery, electric motor, power electronics) need to be maintained within their optimal operating temperature range. TMS is responsible for distributing, transferring and dissipating heat across components under various load and environment conditions. The TMS systems of EVs manage the heat generated by different components either independently or in an integrated manner. Stand-alone TMSs operate independently, which helps identify and improve heat transfer within a particular system, but doesn't account for energy flows between different subsystems[5]. As batteries and motors become more powerful, these separate systems can struggle to meet cooling and heating needs. In contrast, integrated thermal management systems combine multiple subsystems, allowing heat to be distributed properly. This makes them a growing trend in EVs' TMS design since they improve overall energy efficiency and system performance.

Vehicle manufacturers are protective of the internal design of their systems to protect their research and development from competitors. Detailed information on TMS architecture is deficient, with most knowledge available coming from academic research or supplier information. Therefore, the exact layout of the system used in the thesis work cannot be shared. Instead, a modified version is presented in figure 2.1 to reflect key principles.

This TMS is divided into two different temperature circuits: the low and medium temperature circuit. The separation helps manage different components within the thermal system better, improving efficiency and ensuring each part operates within its ideal temperature range.

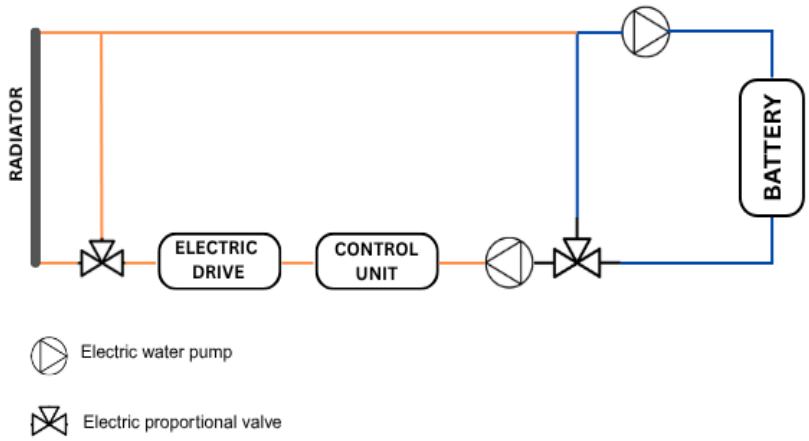



**Figure 2.1:** Block diagram of the TMS illustrating the main components and their interconnections

The low temperature (highlighted in blue in figure 2.1) is responsible for the thermal management of the battery. This circuit's temperature is maintained between $30 - 40°C$, matching the optimal operating range. The medium temperature circuit (highlighted in yellow in figure 2.1) contains the electric motor, inverter, charger, radiator and other low-voltage electronics. This circuit operates around $40 - 70°C$.

The coolant is a glycol-water mixture that circulates through the TMS, absorbing heat from the components and releasing it through radiators. The glycol helps prevent it from freezing or evaporating too easily, which keeps the viscosity more stable and makes heat exchange more effective across a range of temperatures.

TMS includes multiple components that help control the temperatures across the vehicle. Some of the key ones will be briefly described below, without going into too much detail.

Valves are used to direct and manage coolant flow through the TMS. Several types are used in thermal systems, but the one-way check valve and the three-way proportional valve are among the most common. The one-way check valve allows flow only in one direction and prevents backflow, acting as a safety feature in the TMS. The three-way proportional valve can redirect coolant flow to connect parts of the circuit or bypass components. Unlike on/off valves that can only be fully opened/closed, it can split the flow between two paths based on an electronic control signal, allowing precise regulation[11]. There are two common types of three-way valves: L-port and T-port. L-port valves direct flow between two of the three ports and are used for switching between two outputs. T-port valves can connect all three ports and are useful for combining flow paths. Figure 2.2 illustrates various configurations of the T-port valve.

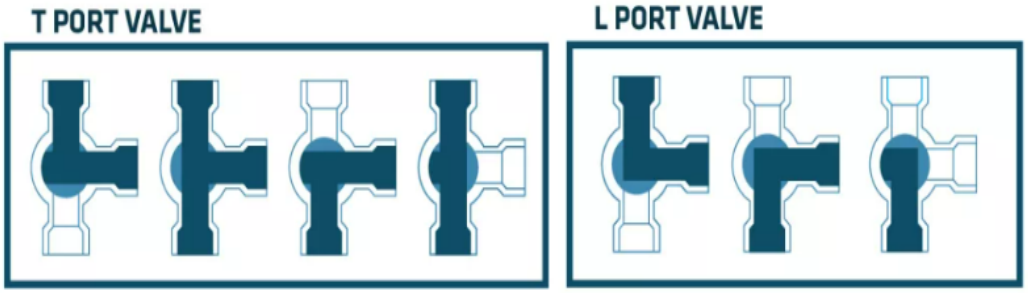



**Figure 2.2:** Possible configurations of the T-port and L-port valves[12]

A pump is a device that uses energy to transport fluids through the system. In a TMS, they are very important, as they help circulate coolant to keep the battery, motor and electronics within their operating temperature(s). There are different types of pumps, but centrifugal pumps are the most common because they are efficient, reliable and simple[13].

The electric drive includes the motor, inverter, charger and other electronics, all of which generate heat and need cooling. Electric motors convert electrical energy into mechanical based on electromagnetic principles. As current flows through the windings, heat is generated due to electrical resistance and cooling is needed to prevent damage and maintain performance. The inverter converts the DC power from the battery into AC power required for the electric motor [14]. It contains sensitive power electronics that must be kept within safe operating temperatures to ensure proper function and avoid damage.

The radiator is a heat exchanger that transfers heat from the coolant to the air, typically through fins on its surface. When the vehicle is moving, airflow is provided, but when it's stationary, a fan helps maintain airflow the keep the system cool [15].

The battery pack stores electrical energy in the EV and supplies direct current to the motor and other systems. Typically, it's made up of lithium-ion cells, which are sensitive to temperature. Therefore, keeping them within their optimal temperature range is crucial to ensure the safety and long lifespan of the vehicle.

## 2.2 System Identification

System identification is the process of developing mathematical models of dynamical systems based on observed input-output data. The process of the derivation of mathematical models of dynamic systems typically differentiates between theoretical and experimental modeling. Theoretical modeling involves obtaining the model by deriving equations from physics (such as balance or interconnection equations). This approach often results in too complex/complicated models that need to be simplified. On the contrary, experimental modeling relies purely on measurements of the inputs and outputs of the system. This is a flexible approach that can be applied

to diverse systems; however, these models do not describe the internal structure of the system. These input-output models are approximations and are still sufficient for many areas of application[9].

In practice, the identification is rarely completely theoretical or completely experimental. The choice of modeling approach depends on the availability of prior knowledge, the required model accuracy, and the desired level of interpretability.
Generally, there exist three types of models to choose from:

1. White-box models are entirely based on well-established principles and physical laws. They assume that relations within the system are known and can be described analytically. They can be difficult to construct when not all relations are fully measurable or understood.
2. Gray-box models are a hybrid approach that combines physical knowledge with measurement data. They use known relationships (where available) and estimate unknown parameters, making them an effective choice when incomplete knowledge is available.
3. Black-box models rely purely on data with no knowledge assumptions on the system structure. Usually, these models are obtained with the use of machine learning or statistical methods(neural networks and regression, for example). They can model highly complex dynamics but are more prone to noise and overfitting and often lack interpretability[17].

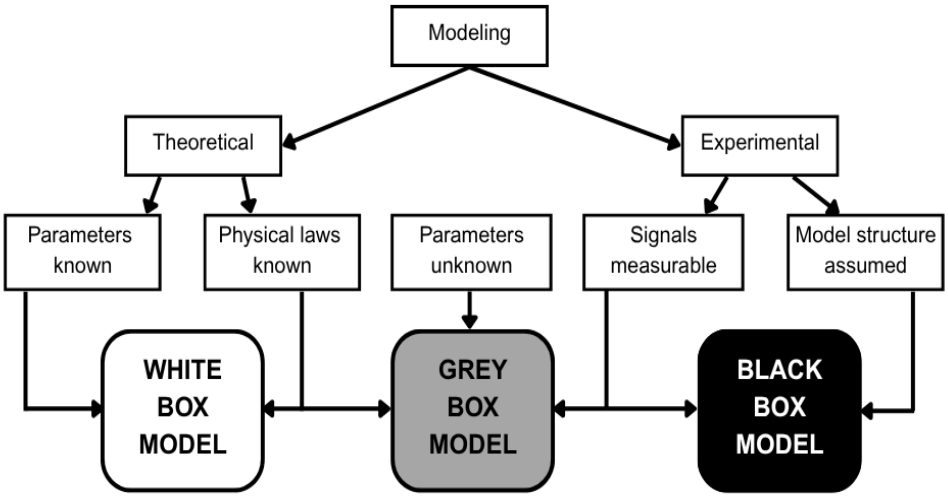



**Figure 2.3:** Diagram presenting classification of mathematical models

Figure 2.3 presents a diagram of comparison of the different modeling approaches discussed. Even though the white-box approach can deliver more information about the system, experimental analysis has found ever-increasing attention over the past 50 years. Some of the main reasons are the reduced complexity, greater precision and the effort/time ratio that is in favor of the experimental analysis. As more industries move to cloud-based systems, data-driven methods provide a scalable and efficient way to build models that are accurate, simple, and yet interpretable[17].

System identification procedure involves three basic entities:

1. Data. The data is usually recorded during an identification experiment that is user-designed. The user determines what signals will be measured and when. The quality of data significantly influences the outcome of the modeling.
2. A set of candidate models. The set is obtained by specifying within which collection of models one will be looking for a suitable one. Choosing a suitable model structure is the most challenging part that requires a combination of a priori knowledge and engineering insight.
3. A rule by which candidate models can be assessed using the data. The assessment is usually based on how the models perform when they attempt to reproduce the measured data[16].

Besides the modeling approach, defining the process structure is another important step. The process structure specifies how many inputs and outputs the system has, whether it is a single input single output(SISO) or multiple input multiple output(MIMO) process. The relationship between the input and the output in SISO systems is direct, making the modeling process simpler. On the other side, inputs and outputs of MIMO systems interact with each other, which increases their complexity[9]. The interactions require more advanced modeling techniques and proper handling of these variable interrelations to create consistent and trustworthy models.

The definition of process behavior involves determining whether the process can be represented by a linear or nonlinear model. Linear system identification is applicable when the superposition principle is satisfied. With this assumption and when using a gray-box model, some of the established model structures with an input-output representation are: ARX (autoregressive with exogenous input), ARMAX (autoregressive moving average with exogenous input) and OE (output error) models. Parameter estimation for these models is most often performed using least squares (LS) or maximum likelihood estimation (MLE). Furthermore, some structures represent models using a linear state-space representation, but they will not be discussed here. Black-box models that use an input-output representation are usually obtained via neural networks and Gaussian processes[17]. While linear models are easier to implement and analyze, and they often provide reasonable approximations around a specific operating point, the limitation lies in their inability to capture complex dynamics properly.

On the other hand, nonlinear system identification is crucial as most real-world systems have inherently nonlinear dynamics where linear assumptions are not sufficient to represent them. When developing nonlinear models it is necessary to select nonlinear function(s) that accurately describe the dynamics. For grey-box approaches with input-output representations, widely used methods include nonlinear variants of the PEM (prediction error model), Kolmogorov-Gabor polynomials, and block-oriented models. For parameter estimation in this case, various methods, including nonlinear least squares (NLS), gradient-based or Bayesian optimization techniques, can be used. They will not be further discussed in this thesis either. Black-box methods use

model structures that work with both state-space and input-output representations. These include neural networks, fuzzy models, machine learning techniques, Gaussian processes, and Koopman-based methods[17]. Although the machine learning and fuzzy models and neural networks are effective in capturing complex nonlinear process dynamics, they do not provide explicit mathematical equations describing the process. Instead, they inherently operate as state-space or input-output representations, which limits their use in applications such as control design. Since the aim of the identification for this thesis is to get the control model, these methods are not suitable. However, alternative approaches to derive mathematical models in the form of explicit equations can be Koopman-based approaches and a method called SINDy (Sparse Identification of Nonlinear Dynamics). SINDy is one of the most recent contributions with a lot of potential due to its ability to identify sparse and interpretable model structures. It chooses the most relevant functions from a large library of functions and provides a model that accurately captures the underlying process dynamics.

Nevertheless, a key challenge in all system identification methods remains the choice of suitable function structures. So far the approaches relied heavily on prior knowledge or trail and error procedure, which is a highly iterative process. No general method that delivers appropriate function structures based on data exists currently. This issues demonstrate the necessity for a systematic approach to address the limitations and ensuring computational efficiency.

## 2.3 SINDy

Sparse Identification of Nonlinear Dynamics (SINDy) represents a framework used to identify governing equations of a dynamical system directly from time-series data. Unlike black-box models, SINDy generates explicit and interpretable mathematical models, suitable for further control and analysis purposes. Furthermore, it can capture nonlinear dynamics efficiently without requiring a predefined model structure and it facilitates the discovery of governing equations in various domains, from biomedical engineering to control applications.

The key idea is to assume that most physical systems have only a few relevant terms that define dynamics[18]. This means that those equations are sparse in a high-dimensional nonlinear function space. Under the assumption, it is possible to identify interpretable models from high-dimensional and complex datasets. Figure 2.4 visualizes this.

In general, a nonlinear dynamical system is in the following form:

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{2.1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ denotes the state vector at time t, and $\boldsymbol{f}(\boldsymbol{x}(t))$ represents the dynamic constraints that define the equations of motion of the system. The function

f is sparse in the space of possible functions, having only a few terms.

In order to determine/approximate function $\mathbf{f}$, two matrices are needed: one representing measurements of the state $\mathbf{X}$ and one with either measurements or numerical approximation of their derivatives $\dot{\mathbf{X}}$. This data is sampled at several times $t_1, ..., t_m$ and should be arranged as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^\mathsf{T}(t_1) \\ \vdots \\ \mathbf{x}^\mathsf{T}(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & \cdots & x_n(t_2) \\ \vdots & \ddots & \vdots \\ x_1(t_m) & \cdots & x_n(t_m) \end{bmatrix} \tag{2.2}$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^\mathsf{T}(t_1) \\ \vdots \\ \dot{\mathbf{x}}^\mathsf{T}(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \cdots & \dot{x}_n(t_m) \end{bmatrix} \tag{2.3}$$

After this, a library of candidate functions $\Theta(\mathbf{X})$ is constructed. It may include polynomials, constants, trigonometric or inverse terms and there is huge freedom in choosing the members. With the assumption that only a few nonlinearities are active in each row, a sparse coefficient matrix $\Xi = [\xi_1, \xi_2, \ldots, \xi_n]$ needs to be computed so that it holds:

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi \tag{2.4}$$

Each column $\xi_i$ of $\Xi$ represents a sparse vector of coefficients that determines which terms are active for the corresponding component $x_i$ of the state. When $\Xi$ is determined, a model of each row of the system equations can be constructed as $\dot{\mathbf{x}}_i = \mathbf{f}_k(x) = \Theta(\mathbf{x}^T)\xi_i$. Here $\Theta(\mathbf{x}^T)$ represents a vector of symbolic functions of elements of $\mathbf{x}$.

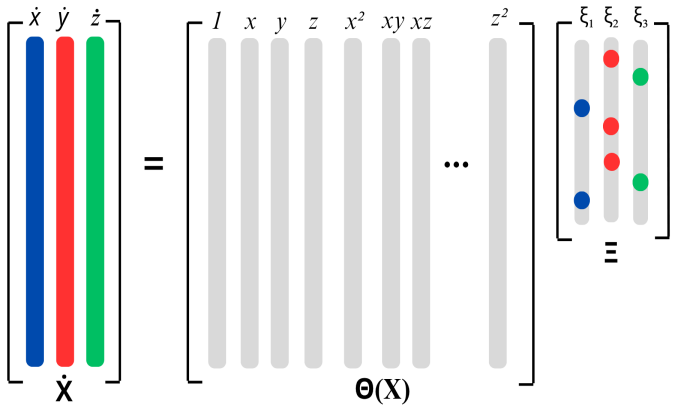



**Figure 2.4:** Illustration of the SINDy formulation

The SINDy framework formulates the task of identifying governing equations as a sparse regression problem in order to isolate only the most essential terms required to accurately represent the data. The two most common methods for solving this problem are Least Absolute Shrinkage and Selection Operator (LASSO) and Sequential Thresholded Least Squares (STLSQ).

LASSO is a regression technique that minimizes the prediction error while enforcing sparsity in the model by introducing an $\ell_1$-norm penalty to the loss function. This penalty term pushes many coefficients to exactly zero, effectively selecting only the most important features. The sparse regression problem in SINDy can be formulated as an optimization task:

$$\boldsymbol{\Xi}^* = \arg\min_{\boldsymbol{\Xi}} \left\| \dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi} \right\|_2^2 + \lambda \|\boldsymbol{\Xi}\|_1 \tag{2.5}$$

where

- $\| \cdot \|_2^2$ minimizes the reconstruction error of the dynamics,
- $\|\Xi\|_1$ promotes sparsity in the coefficient matrix by penalizing the number and magnitude of active terms,
- $\lambda$ is a regularization parameter that controls the trade-off between sparsity and accuracy.

STLSQ is an iterative sparsification technique that first uses the least squares (LS) solution to obtain an approximate solution of $\Xi$. Afterwards, a predefined threshold is applied, setting coefficients with magnitudes below this threshold to zero. The process is repeated, recalculating the remaining nonzero coefficients and updating the model structure until convergence is achieved. This approach is well-suited for deriving interpretable models that still describe the essential complex dynamics of a nonlinear system.

The optimization problem for STLSQ at each iteration is:

$$\min_{\boldsymbol{\Xi}} \left\| \dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi} \right\|_2^2 \tag{2.6}$$

where

$$\Xi_{ij} \leftarrow 0 \quad if \quad |\Xi_{ij}| < \lambda \tag{2.7}$$

SINDy with control[19] is a slightly modified version of SINDy that takes into consideration measurements of the input signals $\mathbf{u}$ when creating the library of candidate nonlinear functions $\Theta$ :

$$\Theta(\mathbf{X}, \mathbf{U}) = \begin{bmatrix} 1 & \mathbf{X} & \mathbf{U} & (\mathbf{X} \otimes \mathbf{X}) & (\mathbf{X} \otimes \mathbf{U}) & (\mathbf{U} \otimes \mathbf{U}) & \cdots \end{bmatrix} \tag{2.8}$$

where $\mathbf{X} \otimes \mathbf{U}$ defines the vector of all product combinations of the components in $\mathbf{x}$ and $\mathbf{u}$. Once again, the recommended strategy is to start with low-order polynomials and then increase the complexity and order of the library until an accurate model is obtained. Equation (2.4) then becomes:

$$\dot{\mathbf{X}} = \Theta(\mathbf{X}, \mathbf{U})\Xi \tag{2.9}$$

The candidate library is the only thing that changes in the optimization problem formulation (see Eq. (2.5)). Figure 2.5 below visualizes this formulation.

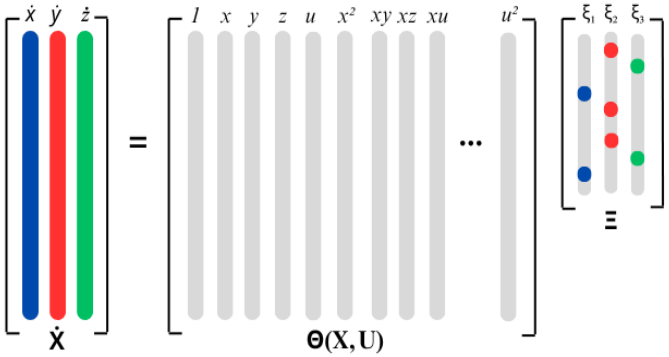



**Figure 2.5:** Illustration of the formulation for SINDy with control

## 2.4 Model Predictive Control

Model Predictive Control (MPC) is a control technique in which the control action is obtained by solving an optimal control problem(OCP) at each sampling instant. The core of MPC is the receding horizon idea that works as follows: at each sampling time, the controller uses current input/output measurements and state values to predict, over a finite horizon, how the system will evolve. Then it picks the control sequence that performs best in terms of the specified objective or cost function. Only the first element in the control sequence is applied to the process while the rest is discarded. This process is repeated at the next sampling time with updated measurements and the horizon moves one step forward[21]. MPC is applicable to MIMO processes and is able to handle constraints and actuator limitations[22]. However, one drawback is that the computational complexity increases rapidly with the size of the system model and the length of the prediction horizon, making the problem very computationally heavy.

An MPC problem can be formulated as:

$$\min_{u} \quad J(x_k, u_k) \tag{2.10}$$

$$\text{subject to} \quad x_{k+1} = f(x_k, u_k) \tag{2.11}$$

$$x_k \in \mathcal{X} \tag{2.12}$$

$$u_k \in \mathcal{U} \tag{2.13}$$

where

- $x$ are the states of the system,
- $u$ are control inputs,
- $J$ is the cost function,
- $f(x_k, u_k)$ is a function that captures the dynamics of the system,
- $\mathcal{X}$ is constraint set for the states of the system,

- $\mathcal{U}$ is constraint set for the inputs to the system.

The cost function usually penalizes deviations from the desired reference trajectory and the use of control effort. It serves as a metric for the optimization algorithm to minimize.

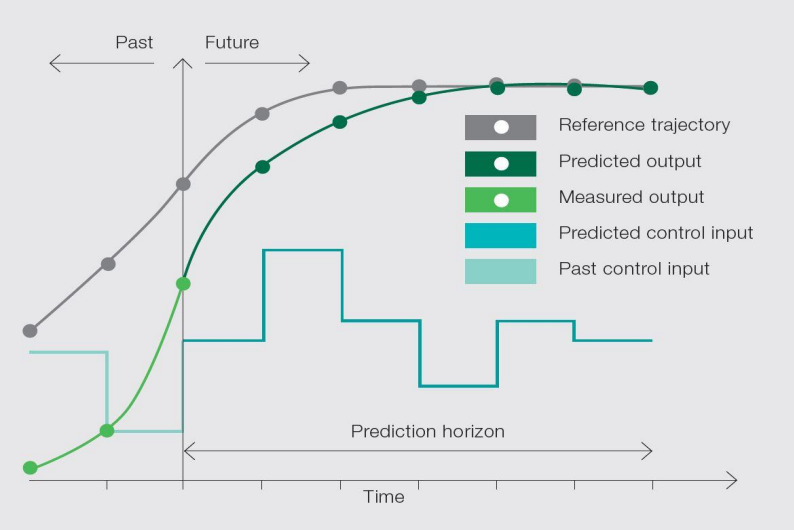



**Figure 2.6:** Visualization of MPC, showing past and future trajectories[23]

The dynamics of the system are typically described by continuous-time equations. Since the controllers are almost exclusively implemented through a computer by sampling variables and applying control inputs at discrete time intervals, the continuous model needs to be discretized. Runge-Kutta 4(RK4) is one of the most widespread integration methods as it presents a good trade-off between computational complexity and accuracy. The RK4 method has four stages and the integration order of the scheme is four[25]. The method is formulated as:

$$x_{k+1} = x_k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.14}$$

where the intermediate steps are calculated as:

$$\begin{aligned} k_1 &= f(x_k, u_k) \\ k_2 &= f\left(x_k + \frac{\Delta t}{2} k_1, u_k\right) \\ k_3 &= f\left(x_k + \frac{\Delta t}{2} k_2, u_k\right) \\ k_4 &= f(x_k + \Delta t \cdot k_3, u_k) \end{aligned} \tag{2.15}$$

## 2.5 Nonlinear Model Predictive Control

Nonlinear model predictive control(NMPC) works by repeatedly solving finite-horizon optimal control problems at each sampling instant based on a nonlinear model of the plant. As already stated, the goal of the OCP is to determine what is the best sequence of control action based on the objective function and some constraints. However, NMPC is harder to apply in industry due to the challenges of guaranteeing a global(or sufficiently good) solution to the optimization problem within the real-time requirements. The nonlinear programming problem(NLP) problem can have multiple local minima and demands a greater number of computations at each sample, with no guarantees on the solution[24].

The numerical solution of the OCP in NMPC is carried out by solving an NLP, which is the discretized and parameterized form of the optimization problem. The OCP is typically rewritten into the standard form of NLP:

$$\begin{aligned} &\min_{w \in \mathbb{R}^n} f(w) \\ &\text{subject to} \\ &g(w) = 0, \\ &h(w) \leq 0 \end{aligned} \tag{2.16}$$

where:

- $f$ is objective/cost function,
- $g$ is the vector of equality constraints,
- $h$ is the vector of inequality constraints,
- $w$ is the vector of optimization variables.

Even though numerical solutions can be found using the indirect methods, focus is placed to direct methods that are more promising[24]. Performing long integrations of nonlinear system dynamics, usually needed to discretize continuous OCPs and solve them as NLPs, can introduce large numerical instabilities. Direct multiple-shooting method solves this by discretizing control input *u(t)* in shorter time intervals, and for each interval the ODE is solved separately. Additional constraints are added to ensure the continuity between the intervals and the problem of piecing the trajectories together is left to the NLP solver[25].

## 2.6 CasADi

CasADi is an open-source software tool designed for numerical optimization and optimal control, particularly involving differential equations. It provides a symbolic framework that allows formulation and manipulation of mathematical expressions, efficient generation of derivatives through algorithmic differentiation, setting up and solving systems of ordinary differential equations (ODEs) or differential-algebraic equations (DAEs). Furthermore, CasADi presents a powerful tool for

formulating and solving both nonlinear programming (NLP) and optimal control problems (OCP). It is accessible across multiple programming languages, including C++, Python, and MATLAB/Octave. There are several NLP solvers interfaced with CasADi. The most popular one is IPOPT, an open-source primal-dual interior point method. Others, that require the installation of third-party software, include SNOPT, WORHP and KNITRO. IPOPT is especially effective for solving general nonlinear programming problems(NLP) and it implements an interior point line search filter method that aims to find a local solution of NLP[20].

# 3
# Methods

## 3.1 Building the control model

This section outlines how the process of system identification was carried out in the thesis, covering methodology and steps taken to develop a model from recorded data.

Data used to develop a model for the TMS was obtained in a simulation environment using GT-SUITE by VCC's team. There were almost 700 cases with approximately 50 variables describing the system. Each case simulated the response of the TMS for different actuator speeds and valve positions. In the simulations, the ambient temperature was constant during one case and there were two scenarios: hot climate($35°$) and cold climate($-15°$). The drive cycle used in the simulations is the Worldwide Harmonized Light Vehicles Test Cycle (WLTC). WLTC is a standardized drive cycle that simulates realistic vehicle operating conditions and includes urban, suburban, and highway driving segments.

### 3.1.1 Data preparation

First step before the identification procedure is data analysis and determining what variables will be used for model development. Using the majority of them would result in a highly complex model and very long equations. After careful consideration, the decision has been made to use the temperatures and the mass flows as system states. The choice is motivated by the fact that temperatures are inevitable as states in a thermal system, while the mass flows of the coolant describe the heat transport through the system. Furthermore, by controlling the mass flows in the system, the temperature can be regulated as well.

One observation made from the system layout and behaviour is that there is a fixed relationship between three of the flows, which can be represented as:

$$\begin{aligned} \dot{m}_1 = \dot{m}_2 + \dot{m}_3 \qquad \dot{m}_2 = a \cdot \dot{m}_1 \qquad \dot{m}_3 = b \cdot \dot{m}_1 \\ \dot{m}_1 = (a+b) \cdot \dot{m}_1 \end{aligned} \tag{3.1}$$

Additional analysis confirmed that throughout all the cases, the assumption $a+b = 1$ holds. This allows the use of only one of the flows instead of three of them and reduces the number of variables. Value of $a$ is determined to be $a \approx 0.48$.

No other simplifications were identified, so the initial approach was to use eight temperatures and eight mass flows as state variables describing the system. Due to the initial goal of the identification being to obtain the control model, to be used for implementation of NMPC that will minimize energy use by the actuators, the electrical powers of pumps were also added as states. The intention behind it is the simplification of formulating and choosing an appropriate cost function.

Selection of inputs of the model was pretty straightforward: the inputs are valves $V_1, V_2, V_3$ and two pumps $P_{ED}$ and $P_{Bat}$. One additional variable that was needed is $T_{amb}$, which was treated as a measurable and known external variable, assumed to be constant over time.

In addition, the data was plotted and checked for noise. Some jitter that could affect both model fitting and future control application negatively was noticed. After further inquiry, it was confirmed that jitter was caused by numerical issues within the simulation and it can safely be filtered out using a moving average filter.

Moving average(MA) filter is a very common filter because it is very easy to understand and use. Furthermore, it is optimal for a common task: reducing random noise while retaining a sharp step response. The moving average filter operates by averaging several points from the input signal to produce each point in the output signal[27]. In equation form, this is written as:

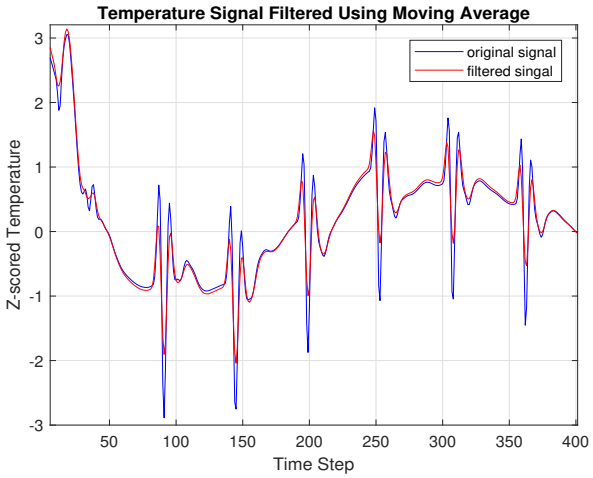
$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j] \tag{3.2}$$

where $x$ is the input signal, $y$ is the output signal, and M is the number of points in the average. Figure 3.1 illustrates the application of the MA filter to one temperature signal and one mass flow signal.

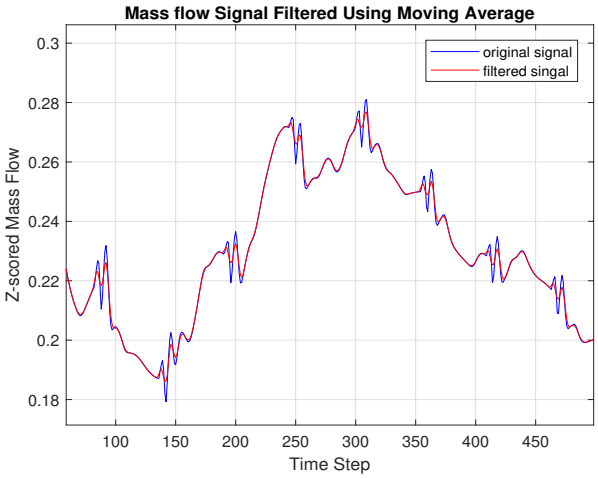
The original data cannot be shown in the report due to proprietary restrictions from VCC. Therefore, all data shown is normalized using Z-score normalization. Z-score normalization is the process of normalizing every value in a dataset such that the mean of all of the values is 0 and the standard deviation is 1[28]. The following formula is used to perform normalization on the data:

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{3.3}$$

where $\mu$ is mean of data and $\sigma$ is standard deviation of data. Another commonly used normalization method is min-max normalization, but it wasn't chosen here because there was an offset between the original and filtered data, caused by the fact that each dataset is scaled relative to its minimum and maximum.




**(a)** Temperature signal filtered




**(b)** Mass flow signal filter

**Figure 3.1:** Input signal before and after applying the MA filter.

Data has been split into training and test sets in an 80/20 ratio. This is done randomly, but paying attention that each case, with all its samples, is kept together as one unit. The chosen tool for the identification process is MATLAB, as it is very convenient for working with matrices. The base was code published as part of the [18], but it was later modified as needed.

### 3.1.2 Numerical differentiation

As previously mentioned in 2.3, a matrix of state derivatives $\mathbf{X}$ is necessary. That information was not included in the simulation data, so a numerical derivative approximation was needed. The simplest method is the central finite difference, which is computed as:

$$\left.\frac{dx}{dt}\right|_i \approx \frac{x_{i+1} - x_{i-1}}{2\Delta t} \tag{3.4}$$

Unfortunately, this method did not provide derivative estimates accurate enough, resulting in SINDy returning trivial dynamics such as straight lines or zero functions and failing to identify a meaningful model. Fortunately, there is an open-source Python library called *pysindy*, designed specifically for SINDy implementation that includes a range of derivative approximation techniques. Some of the methods included and tested are Smoothed Finite Difference, Spline Derivative and Spectral Derivative. Among them, the Spline Derivative proved to be the only method that produced viable results for this dataset. The Spline Derivative approach fits the data with smooth polynomial curves and then performs differentiation on the spline[30].

### 3.1.3 Identification using SINDy

As previously discussed, several methods exist for solving the SINDy problem. In this work, the Sequential Thresholded Least Squares (STLSQ) method was used. STLSQ begins by solving a standard least squares problem to estimate all coefficients, then applies a threshold to eliminate those below a certain magnitude. The process is repeated with the reduced set of terms until convergence, as shown in Algorithm 1. A detailed implementation of the algorithm can be found in Appendix A.1.

**Algorithm 1** Sparse Identification of Nonlinear Dynamics

**Input:** Candidate function matrix $\Theta$, derivative data $\dot{X}$, sparsity threshold $\lambda$, state dimension $n$
Initialize coefficient matrix $\Xi \leftarrow \Theta \backslash \dot{X}$ ▷ Initial least squares solution
**for** $k = 1$ to $10$ **do**
    Identify small coefficients: $smallIndices \leftarrow \{(i,j) \mid |\Xi_{ij}| < \lambda\}$
    Set small coefficients to zero: $\Xi_{smallIndices} \leftarrow 0$
    **for** $j = 1$ to $n$ **do**
        Identify large coefficients: $bigIndices \leftarrow \{i \mid |\Xi_{ij}| \geq \lambda\}$
        Update coefficients by regression:

$$\Xi_{bigIndices,j} \leftarrow \Theta_{:,bigIndices} \backslash \dot{X}_{:,j}$$

    **end for**
**end for**
**return** sparse coefficient matrix $\Xi$

The candidate function library for the TMS model consisted of polynomial terms up to degree two, along with an inverse function. Parameter $\lambda$ is the sparsity threshold used to determine which coefficients are considered significant, since any coefficient with absolute value smaller than $\lambda$ is set to zero. A larger $\lambda$ leads to fewer retained terms (more aggressive sparsification), while a smaller $\lambda$ allows more terms to be included. Values of $\lambda$ in the range 0.001–1 were tested initially.

Even with the different numerical approximation approach, flows remained identified as straight lines in SINDy. Further investigation came to a conclusion that flows had

smaller magnitudes, up to 5000 compared to other variables. Solution for that was rescaling the flow to [l/min] instead of [l/s]. After the rescaling and recomputation of numerical differentiation, mass flow identification in SINDy yielded significantly better results.

To numerically assess the model's accuracy, Root Mean Square Error (RMSE) and Symmetric Mean Absolute Percentage Error (SMAPE) were used. The RMSE is defined as the measure of the differences between values that are predicted by a model and values that are actually observed:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.5}$$

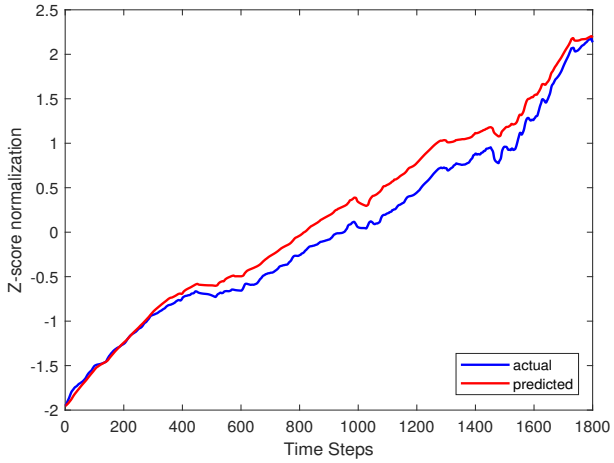
where $n$ is the number of observations[31].

SMAPE is a common metric for evaluating prediction accuracy that measures the relative difference between predicted and actual values, treating over- and under-estimates equally. SMAPE values range from 0% (perfect prediction) to 100% (no similarity)[32]:

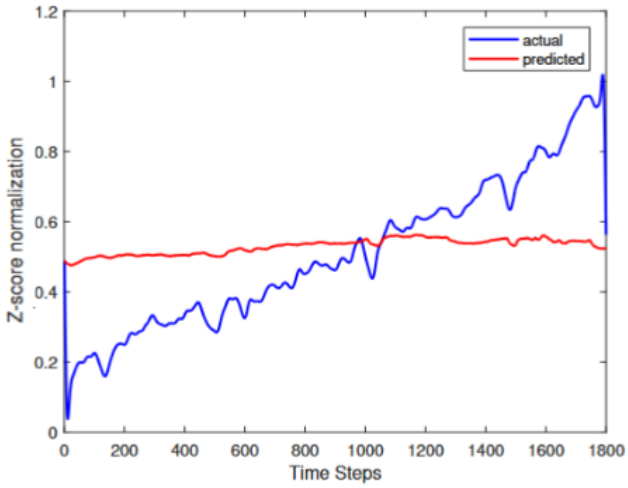
$$\text{SMAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{(|\hat{y}_i| + |y_i|)/2} \tag{3.6}$$

However, with this library of candidate functions, it was not possible to obtain a model that fully captured the behaviour of the TMS with satisfactory accuracy. Here, accuracy refers not only to the numerical closeness but also to the ability of the model to follow the system's trend(i.e., increasing when the measured signal increases and vice versa). Figure 3.2 illustrates examples of both successful and poor trend capturing by the fitted models.

This outcome is not surprising when taking into account the complexity of the system. An alternative approach was to try and identify models of different physical configurations of the TMS, based on the assumption that these would exhibit more consistent behaviour. It should be feasible to implement a switching mechanism between these models at a later stage.




(a) Good trend tracking




(b) Poor trend tracking

**Figure 3.2:** Example showing the model's ability to capture system behaviour

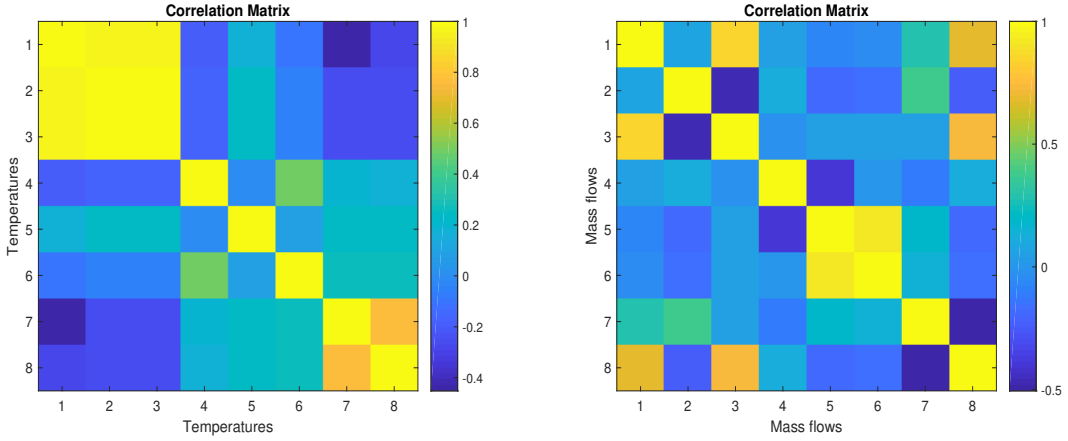
Among four different configurations tested, SINDy managed to provide reasonably accurate models for only two. The better performing one, describing the heat pumping scenarios, was selected as the primary focus for the rest of this thesis. All further attempts on controller development and improvement were done on this configuration.

Model assessment repeatedly failed for this configuration, unfortunately. To address this and obtain a control model that can be used for developing a controller, the state variables were carefully reviewed and analyzed for similarity, in order to reduce the number of states in the model. With fewer states, the model equations are simpler and less computationally heavy, possibly yielding a more practical control model.

### 3.1.4 Simplification of the model

A correlation analysis was performed on both temperature and mass flow measurements for the previously selected TMS configuration. The resulting correlation matrices, shown in figure 3.3, illustrate the pairwise correlation coefficients between the eight measurements for each variable. Dark blue indicates strong negative, while yellow represents strong positive correlation. The diagonal values represent perfect correlation of each variable with itself.
The matrices show that several measurements are highly correlated and can be accurately represented as linear or piecewise combinations of others. In consequence, the redundant variables were excluded, reducing the number of states from eighteen to nine (four temperatures and five mass flows). This reduction is valid only for the chosen TMS configuration and does not generalize to other setups.

Additionally, the pump speed data was scaled prior to analysis. Without scaling, its large magnitude caused it to appear less frequently in the model equations, underrepresenting its influence in the system dynamics.




**(a)** Temperatures **(b)** Mass flows

**Figure 3.3:** Correlation matrices for system variables

Regrettably, even with this simplification, the model assessment results indicated that the obtained equations did not represent the system dynamics with sufficient accuracy. The resulting model structure can't be considered suitable for controller design.
To address this, the next step was generating alternative models with different levels of complexity and accuracy: starting from linear models, including nonlinear models of orders 1 and 2. The performance of these models would then be evaluated and compared. This way, it could be investigated if simpler models that will be less accurate may be more suitable for control purposes. The investigation was carried out using data from the same TMS configuration and reduced model with 9 states. During the analysis, the parameter $\lambda$ was systematically varied and the resulting model was evaluated both visually through variable plots and quantitatively using the RMSE and SMAPE metrics.

## 3.2 Model assessment

To assess the model, it was used in an NLP framework implemented via direct multiple shooting(DMS). The implementation was in MATLAB using the CasADi framework with the IPOPT solver. This setup forms the basis for an NMPC implementation, but in this case it was executed offline and solved only once, for assessment purposes.
The modified version of the code is attached in the Appendix A.2. The equations are replaced with placeholders due to confidentiality restrictions, but the structure allows easy modifications to tailor the code to a specific application.

Besides inserting the system equations, other key elements required for the setup were: time horizon length $T$, number of control intervals $N$, the cost function $L$ and vectors for the initial states and for the constraints.

The cost function $L$ in the initial test was defined for setpoint tracking since the primary control objective of the TMS is maintaining key temperature states within desired values while respecting operational constraints. This was implemented by penalizing the squared tracking error of the battery temperature, as the battery is the most critical component to be maintained within its operational range:

$$L = (T_{bat} - T_{ref})^2 \tag{3.7}$$

$T_{ref}$ was chosen as a constant. Initial temperatures in simulation data were always set equal to $T_{amb}$ and this assumption was retained during model assessment. Initial values for mass flows were also taken directly from simulation data, using values from the chosen configuration.

Optimization variables in this scenario are states $x$, temperatures and mass flows, and inputs $u$, valve positions and pump speeds, which are stacked in the optimization variables vector $w$ in the following way:

$$w = \begin{bmatrix} x_0 & u_0 & x_1 & u_1 & ... & x_{N-1} & u_{N-1} & x_N \end{bmatrix}^T \tag{3.8}$$

Next to be defined are the equality and inequality constraints. The equality constraints are introduced to ensure consistency of the discretized system dynamics: the state obtained from integrating the system over one interval must match with the value of the subsequent state optimization variable. This guarantees that the continuity of the trajectories is preserved across all shooting intervals and is mathematically expressed as:

$$x(i+1) = \tilde{f}(x(i), u(i)) \rightarrow x(i+1) - \tilde{f}(x(i), u(i)) = 0 \tag{3.9}$$

Inequality constraints were formulated as simple box constraints, specifying lower and upper bounds on both the system states and control inputs:

$$x_{min} \leq x_i \leq x_{max} \qquad u_{min} \leq u_i \leq u_{max} \tag{3.10}$$

Input constraints were formed right away, as clear minimum and maximum values exist for both valve positions and pump speeds, reflecting their physical actuation limits. On the other hand, defining bounds for the states was less straightforward, as these depend on operational requirements and safety considerations rather than strict physical limits.

Primary goal of the TMS is to maintain the battery temperature within a suitable range, since the battery is the most critical component operating under the strictest constraints. According to [4], the theoretical operational temperature window of Li-ion batteries is -10°$C$ to 50°$C$, but performance degradation occurs at low temperatures, while aging and safety risks accelerate above 50°$C$. Therefore, it is recommended that the TMS maintains the battery temperatures between 15°$C$ and 35°$C$. Other compontents in the TMS are of a secondary priority and can withstand higher temperature ranges.

The final decision on the applied temperature limits was made based on additional analysis and consideration of the conditions under which the simulation data had been obtained. The overall aim was to relax the constraints as much as possible without compromising safe and reliable operation. Based on these considerations, the battery temperature states were constrained between $-15°C$ and $50°C$. Upper bound on other temperature states was set $70°C$. All mass flows were bounded between 0 l/min and 30 l/min.

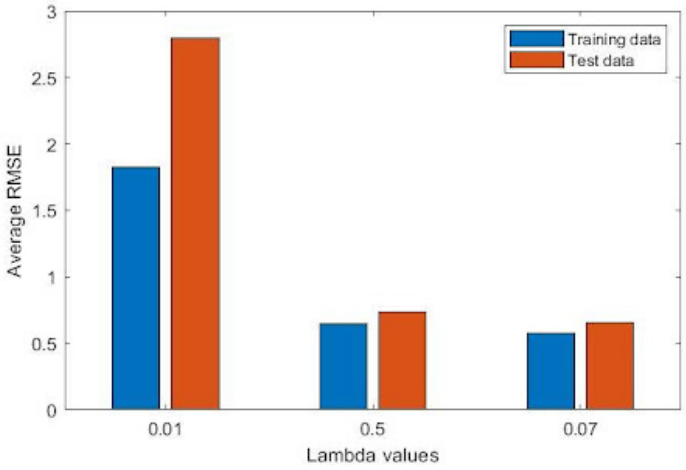
# 4

# Results and discussion

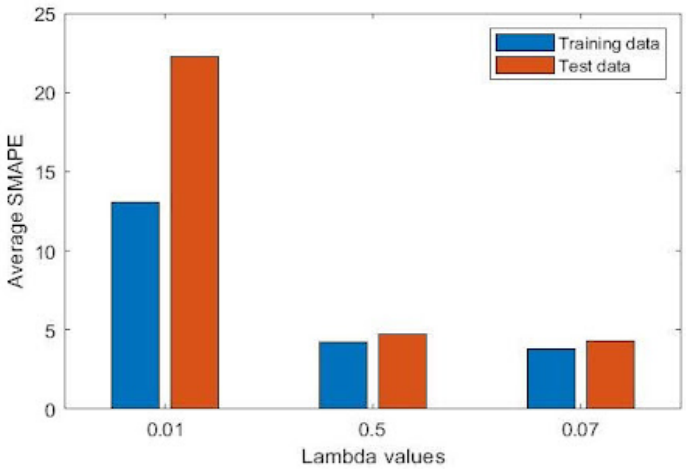
## 4.1 Model identification using SINDy

The first model identified consists of 18 states, which makes presenting both signal plots and evaluation metrics (RMSE and SMAPE) for each state individually impractical. Therefore, the performance assessment was carried out in the following way: first, the predictions obtained from models identified with different $\lambda$ values were plotted with the actual system data and visually inspected. The plots helped analyze the model's ability to capture the system's dynamics. In case the model demonstrated good performance (good trend capturing), quantitative evaluation was carried out by computing RMSE and SMAPE for each state. The detailed results for are provided in Appendix B.1. Since the goal is to select the best performing model, for further assessment, the average RMSE and SMAPE across all temperature and all mass flow states were calculated for each $\lambda$ value. These metrics are visualized as bar charts shown in the figures below. The evaluation is performed this way on both the training and test data to confirm generalization capability.

Figure 4.1 presents average RMSE and SMAPE for temperature predictions across three models obtained by using different regularization parameters. Blue bars correspond to results on training, while red ones correspond to results on test datasets. It is clear that the model for $\lambda = 0.01$ performs significantly worse than the other two. Models for $\lambda = 0.5$ and $\lambda = 0.07$ achieve comparable performance for temperature states. Figure 4.2 illustrates the same metrics for mass flow states. Once again, the errors are largest for $\lambda = 0.01$, but $\lambda = 0.07$ shows lower error across both metrics. Based on this, the model with $\lambda = 0.07$ was selected for further use.

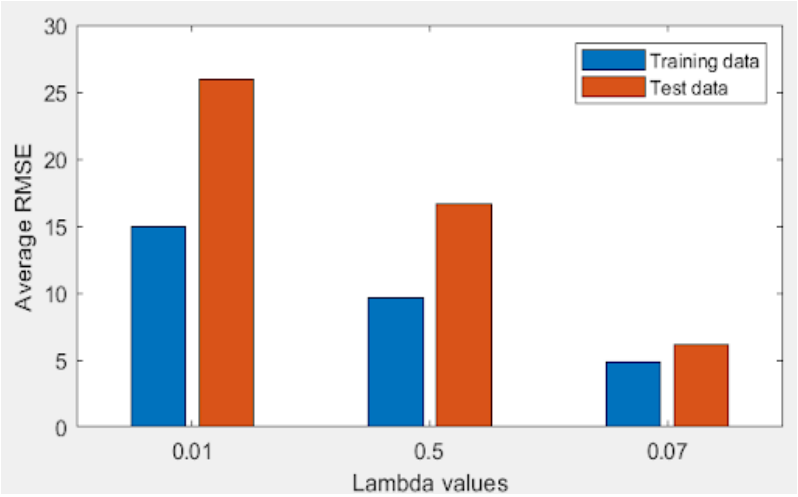



**(a)** Average RMSE

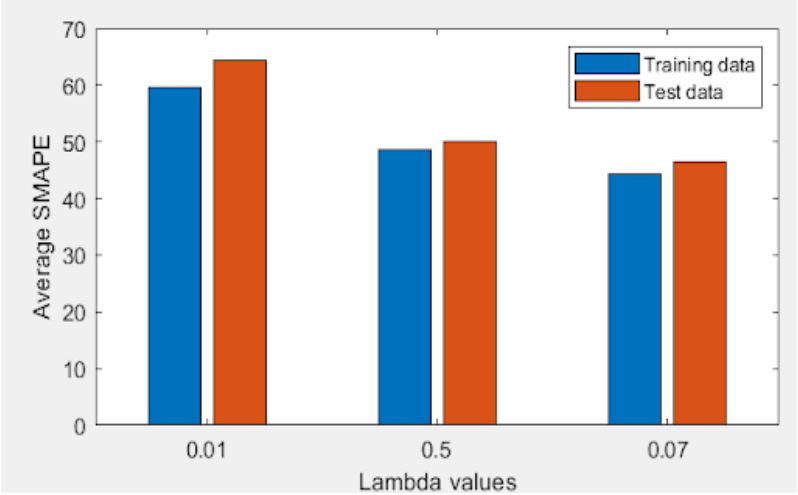



**(b)** Average SMAPE

**Figure 4.1:** Evaluation metrics for temperature predictions in training and test datasets for three $\lambda$ values, model with 18 states



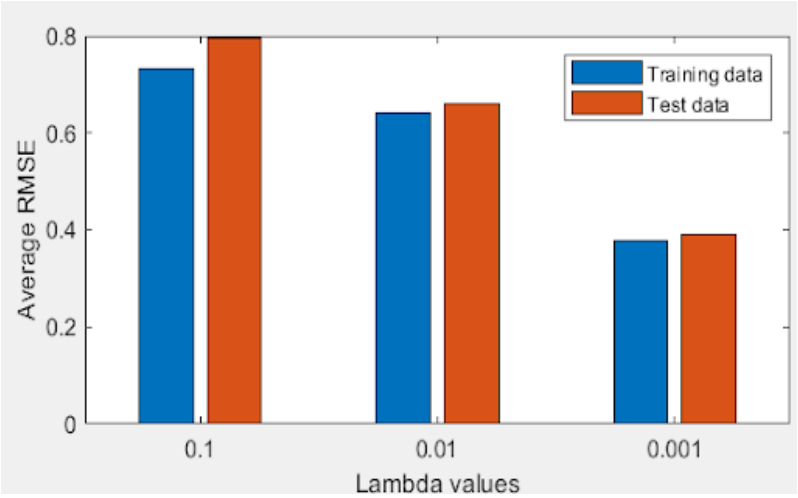

**(a)** Average RMSE



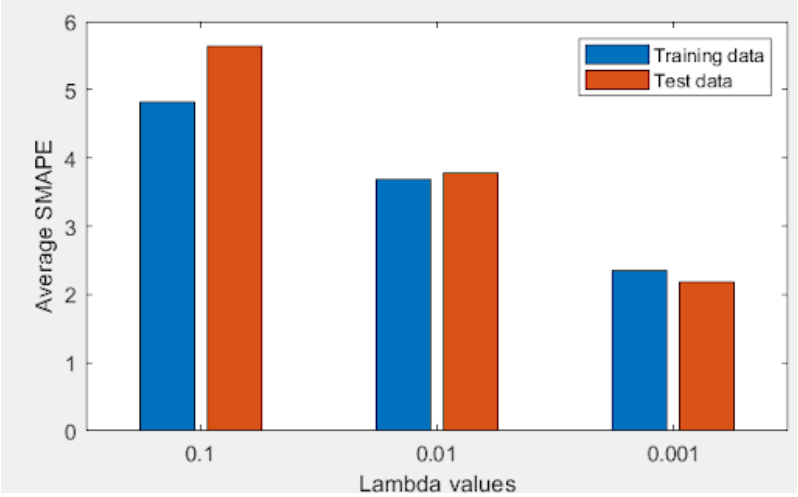

**(b)** Average SMAPE

**Figure 4.2:** Evaluation metrics for mass flow predictions in training and test datasets for three $\lambda$ values, model with 18 states

The same evaluation steps were taken for the reduced model with 9 states. RMSE and SMAPE were computed for both temperature and mass flow states, and the results were compared across different $\lambda$ values. Bar charts for temperatures, shown in figure 4.3, confirm that $\lambda = 0.001$ proved to yield the smallest errors. The same trend was observed for mass flows, figure 4.4. However, the difference between $\lambda = 0.01$ and $\lambda = 0.001$ is relatively minor(around 1% or less on SMAPE plots). A larger $\lambda$ value results in a simpler and less computationally heavy model, so the model obtained for $\lambda = 0.01$ was selected as the final choice.

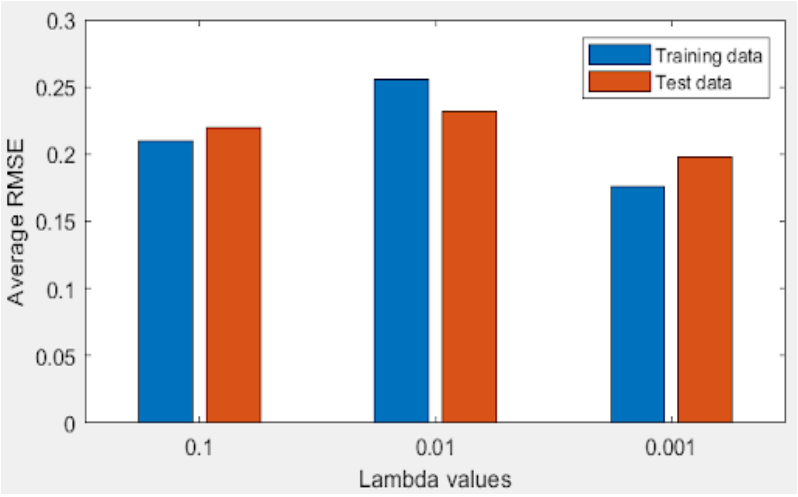



**(a)** Average RMSE

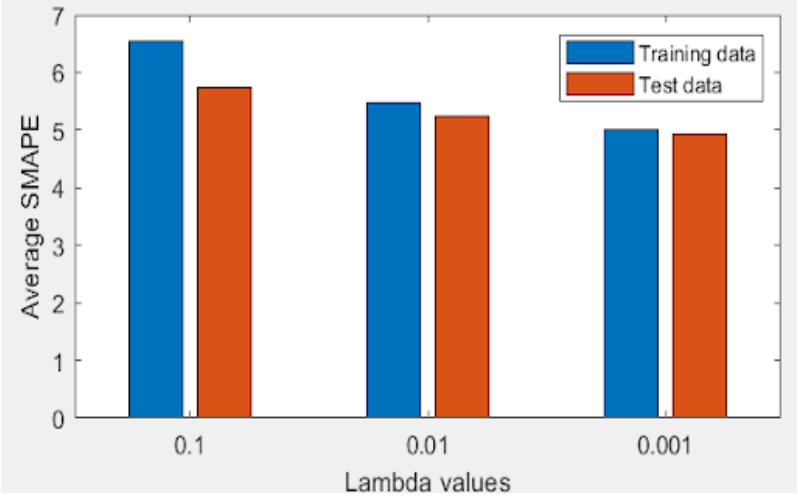



**(b)** Average SMAPE

**Figure 4.3:** Evaluation metrics for temperature predictions in training and test datasets for three $\lambda$ values, model with 9 states




**(a)** Average RMSE




**(b)** Average SMAPE

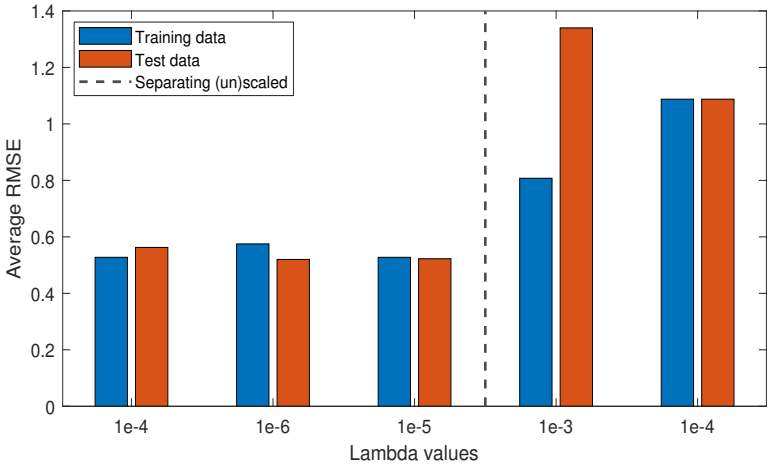
**Figure 4.4:** Evaluation metrics for mass flow predictions in training and test datasets for three $\lambda$ values, model with 9 states

## 4.2 Model comparison

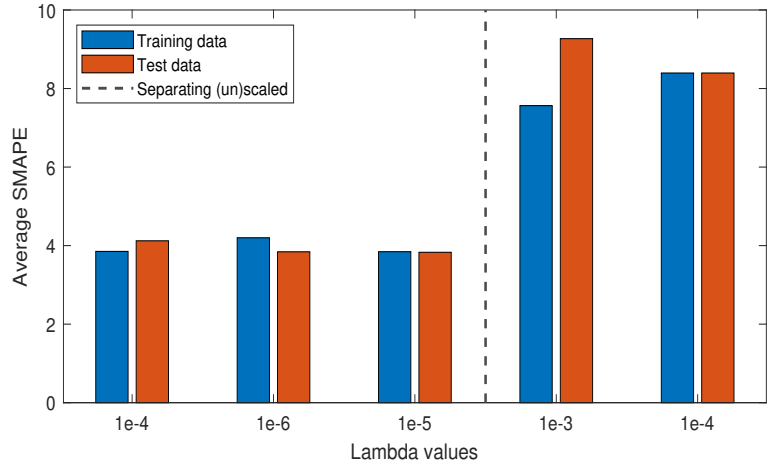
After the unsuccessful assessment of the 18- and 9-state models, alternative models of varying complexity were assessed and the results are shown below. While these models are expected to deviate from the true dynamics of the system, the aim is to try and evaluate if a simpler model would be adequate for controller development.

Initially, the models were identified using the raw dataset, but an issue with the simple linear models arose. The control inputs(pumps speed) appeared only in a couple of equations, which is problematic because a model that lacks influence from control inputs cannot be effectively used for controller development. The assumption was that this behavior may be a result from the larger magnitude of pump speeds, so these variables were scaled to bring their range closer to that of other inputs.

Figures 4.5 and 4.6 show evaluation metrics bar charts for linear model (both temperature and mass flow states, same as for identification results), while figures 4.7 and 4.8 represent the same metrics, just for polynomial model of order 2. The vertical line in all plots is dividing models with unscaled (left of the line) from those with scaled pump inputs (right of the line). For all unscaled cases, the metrics are relatively close across $\lambda$ variations. There are some differences between scaled and unscaled metrics, but they are not drastic. The linear models required a very small $\lambda$ value to prevent all the coefficients from being equal to zero. On average, the temperature errors are slightly higher for the linear models, while for the mass flow signals, the polynomial models give larger errors. Larger $\lambda$ values have been chosen for polynomial models to enforce simpler, short equations, so it is not surprising that those models show higher errors. Exact prediction accuracy is not a concern here since the models are intended for control design comparison.

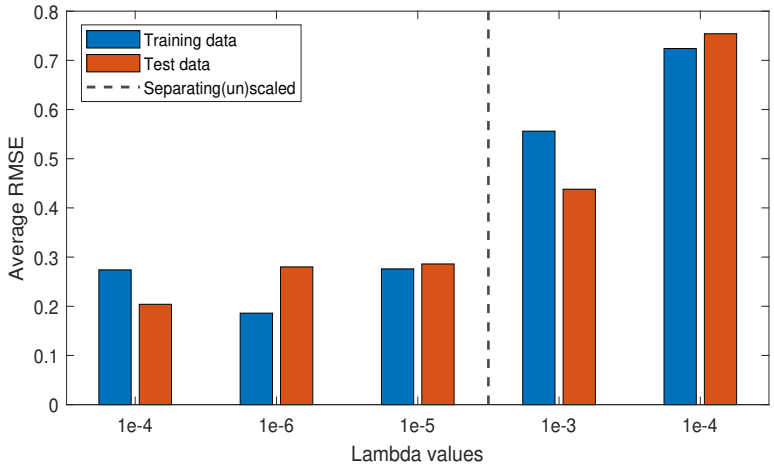



**(a)** Average RMSE

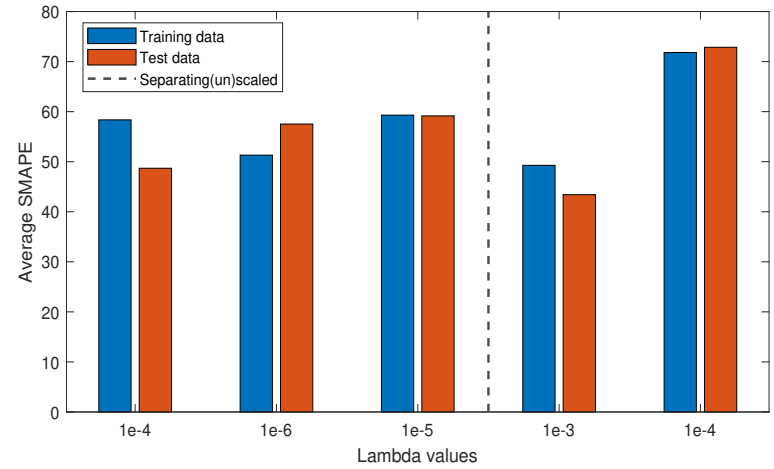



**(b)** Average SMAPE

**Figure 4.5:** Evaluation metrics for temperature predictions in training and test datasets for five $\lambda$ values, linear model

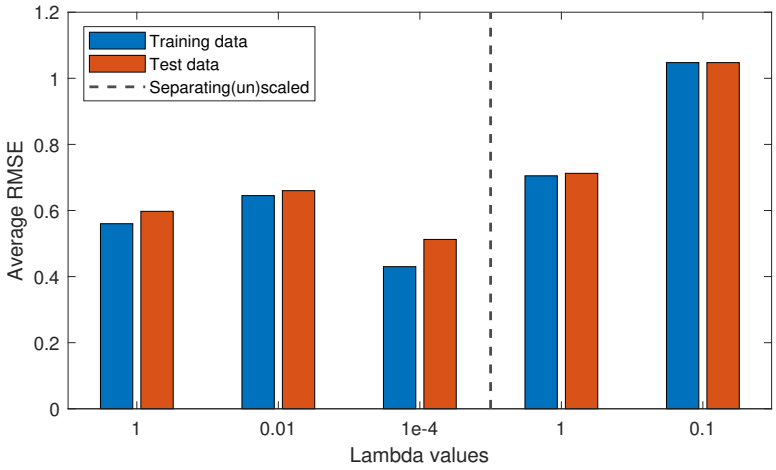



**(a)** Average RMSE

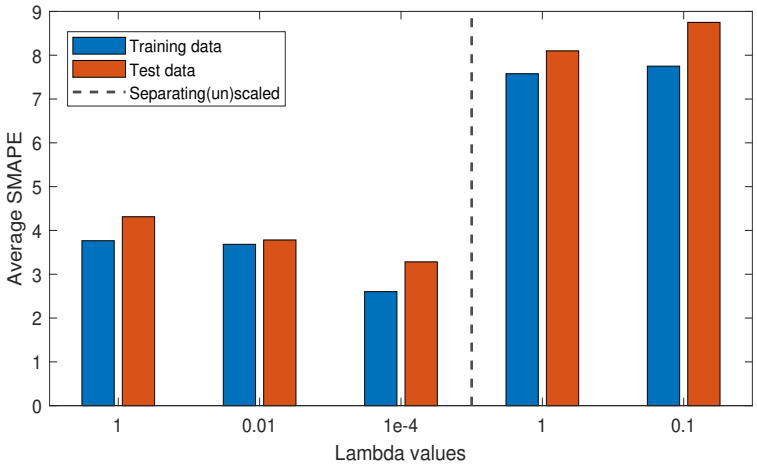



**(b)** Average SMAPE

**Figure 4.6:** Evaluation metrics for mass flow predictions in training and test datasets for five $\lambda$ values, linear model

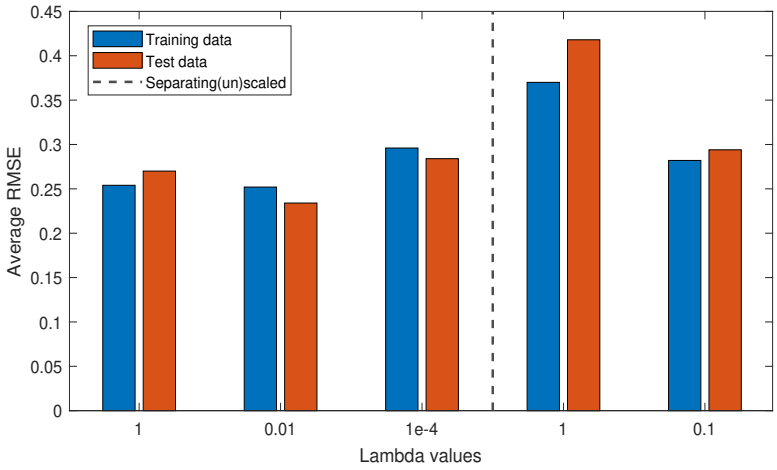



**(a)** Average RMSE

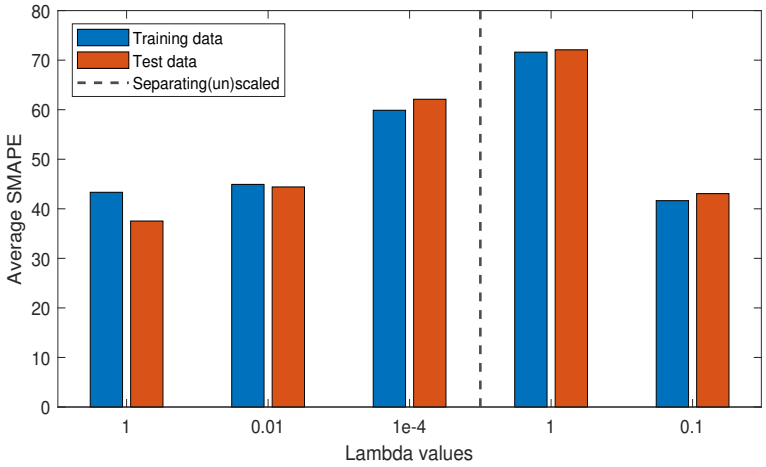



**(b)** Average SMAPE

**Figure 4.7:** Evaluation metrics for temperature predictions in training and test datasets for five $\lambda$ values, polynomial model



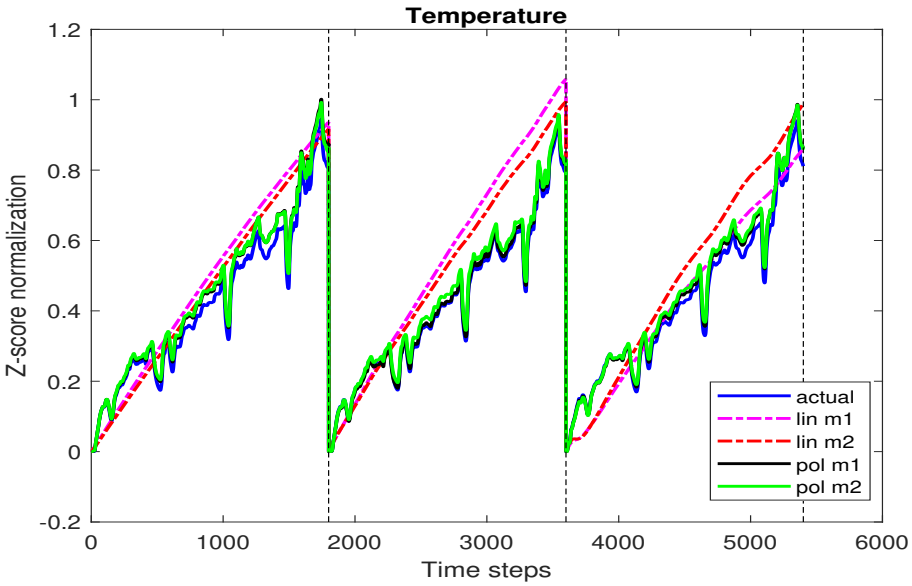

**(a)** Average RMSE



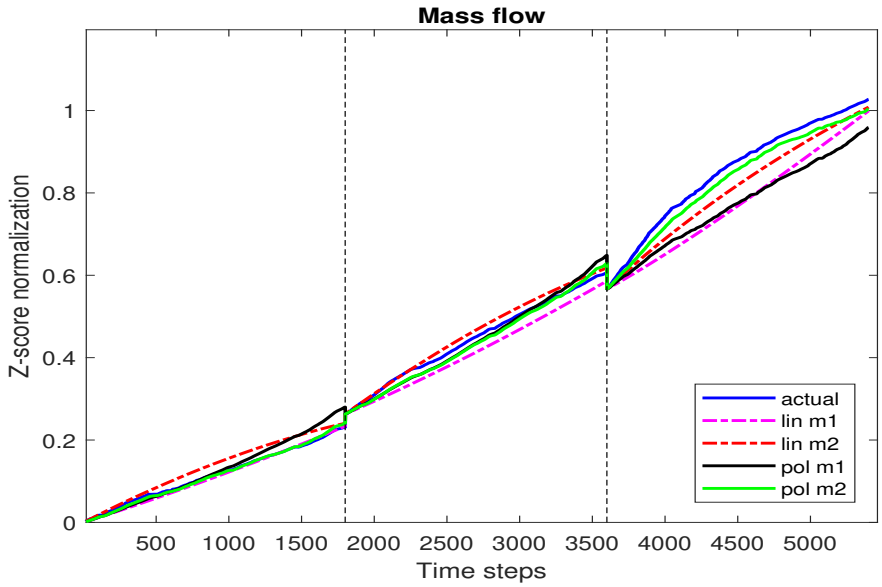

**(b)** Average SMAPE

**Figure 4.8:** Evaluation metrics for mass flow predictions in training and test datasets for five $\lambda$ values, polynomial model

Model assessment was done on models with scaled pump inputs: linear models for $\lambda = 0.001$ and $\lambda = 1e^{-4}$, polynomial models of order 2 for $\lambda = 1$ and $\lambda = 0.1$. Figure 4.9 illustrates an example comparison for one randomly selected temperature and one mass flow state, showing the actual signal and four approximations obtained from the models of different complexity and $\lambda$ values. From figure 4.9 (a) it can be observed that both linear models (1 and 2) can not capture the nonlinear signal behavior (rapid changes and curvatures), as expected. The polynomial models follow the shape more closely, but they do show slight deviations. In figure 4.9 (b), linear models were not too far off from the actual trends.




(a) Temperature signal




(b) Mass flow signal

**Figure 4.9:** Example of model performance for one temperature and one mass flow state under different complexity levels and $\lambda$ values

## 4.3 Model assessment

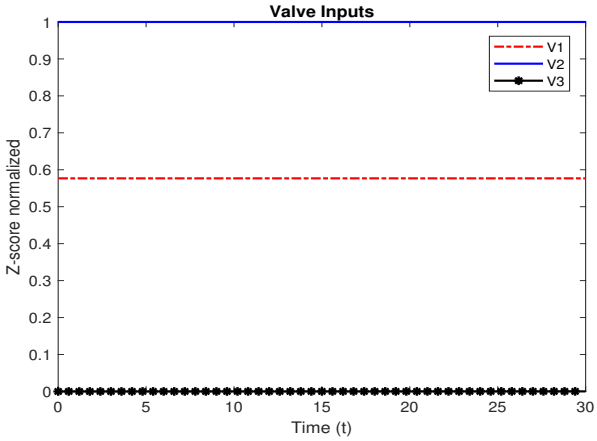
The 18- and 9-states models were assessed using the NLP framework. During the assessment process, three common termination messages from IPOPT appeared:

- Maximum_Iterations_Exceeded: indicates that IPOPT has exceeded the maximum number of iterations specified before finding the optimal solution;
- Restoration_Failed: means that IPOPT tried to recover from an infeasible/problematic step but was not able to find a feasible point that satisfies the constraints;
- Infeasible_Problem_Detected: occurs when it's determined that the problem can't satisfy all constraints and indicates actual infeasibility (or the algorithm is stuck at a locally infeasible point)[34].

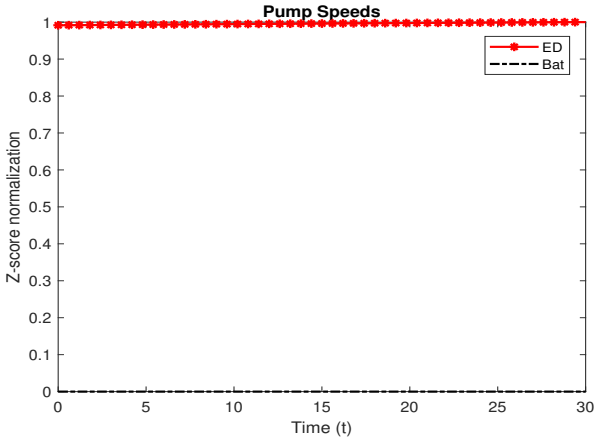
All of these termination messages point to solver failure. Several strategies were attempted to overcome this, including loosening constraints, changing initial state guesses and adjusting the horizon parameters $T$ and $N$. Unfortunately, none of them lead to finding optimal solution. The solver likely got stuck in a local minimum, which is common risk when solving NLP problems.

When assessing models of different complexities, the simpler linear model ($\lambda = 1e^{-4}$) and polynomial model ($\lambda = 1$) were able to yield optimal solutions. However, the solutions were obtained for only a limited set of $T$ and $N$. For $T > 50$ seconds, achieving an optimal solution was impossible and one of the previously described IPOPT termination messages appeared. If $T$ was set to be larger, $N$ had to be at least $2 \cdot T$ to avoid the Restoration_Failed error. This adjustment increased the number of solver steps significantly, causing the algorithm to keep exceeding the maximum iteration limit, even when it was raised to 10000 iterations (the default is 3000, typically sufficient for most NLP problems). This is problematic behavior, as the overall goal of the thesis is to apply the controller to drive cycles that last $\approx 30$ minutes. Solutions valid for horizons this short only provide little value.

Figures below are provided to illustrate the results for both the linear and polynomial models under cold climate conditions ($-15°C$) using $T = 30$ and $N = 50$. For longer $T$/different $N$, the behavior stays the same, the only difference being a stretched time axis(remains under two minutes). These plots serve to display the outputs and highlight why further controller development was not feasible. As shown in figures 4.10 and 4.11, the signals are constants, meaning no actual control takes place. Under such a short horizon, a TMS is not expected to meaningfully influence the system dynamics.

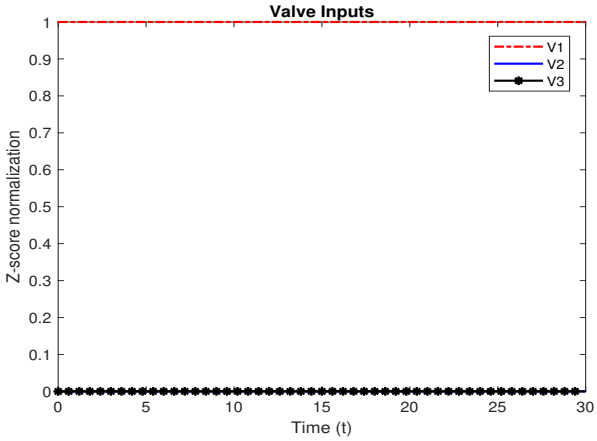



**(a)** Valves signals

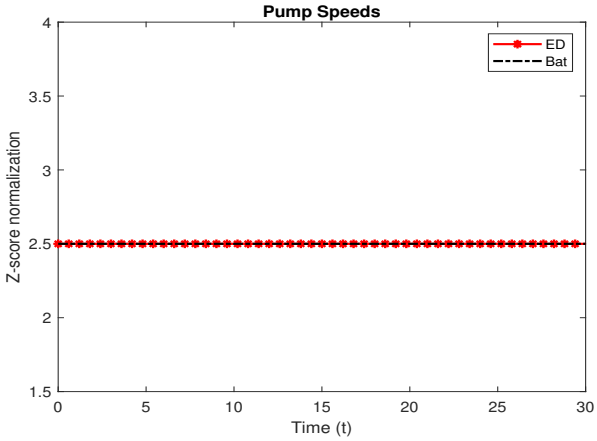



**(b)** Pumps signals

**Figure 4.10:** Actuator input signals under cold climate conditions for a linear model




**(a)** Valves signals




**(b)** Pumps signals

**Figure 4.11:** Actuator input signals under cold climate conditions for a polynomial model

## 4.4 Future work

The grey box modeling approach taken in this work can be considered a dark grey box approach, since it relies more on empirical data than use of physical knowledge. Unfortunately, the performance of these models was not satisfactory for intended application. A better direction for future research would be moving toward more theoretical, physics-based modeling strategies such as white or light-grey box modeling approaches. Potential directions include component-based modeling of key TMS subsystems (battery, electric circuit, pumps) as well as developing thermodynamic and fluid dynamic equations that capture system behavior. The combination of physics-based modeling and data-driven parameter tuning may provide models that are computationally efficient and also more reliable for controller design.

# 5
# Conclusion

Unfortunately, the goal of the thesis was not achieved, as the chosen identification approach did not result in an accurate control model of the TMS that can be used for the controller design. While the simple linear and polynomial models yielded optimal solutions for short prediction horizons, these results were not meaningful given the intended application on drive cycles of 30+ minutes. Larger horizons consistently resulted in solver failures despite attempts to overcome the issue by adjusting constraints and initial conditions. This indicates that meaningful control can not be achieved with this model. Developing a more accurate, physics-based model that better captures the nonlinear thermal dynamics is a promising direction for future research. After improving the model, it will be possible to perform a detailed controller analysis and draw definitive conclusions on the suitability of this approach for the control of a TMS in a BEV.

# Bibliography


[1] P. Wingender et al., "Europe's shift to EVs amid intensifying global competition", *International Monetary Fund*, 2024.

[2] European Environment Agency, "New registrations of electric vehicles in Europe", *EEA*, [Online]. Available: `https://www.eea.europa.eu/en/analysis/indicators/new-registrations-of-electric-vehicles`. [Last accessed: Aug. 22, 2025].

[3] Y. Ding et al., "Automotive Li-Ion batteries: Current status and future perspectives", *Electrochemical Energy Reviews*, vol. 2, no. 1, 2019.

[4] G. Xia, L. Cao, and G. Bi, "A review on battery thermal management in electric vehicle application", *Journal of Power Sources*, 2017.

[5] D. Dan, Y. Zhao, M. Wei, and X. Wang, "Review of thermal management technology for electric vehicles", *Energies*, 2023.

[6] P. Lokur et al., "Modeling of the thermal energy management system for battery electric vehicles", in *Proc. IEEE Vehicle Power and Propulsion Conf. (VPPC)*, 2022.

[7] P. Lokur et al., "Distributed model predictive controller for thermal energy management system of battery electric vehicles", in *Proc. 62nd IEEE Conf. Decision and Control*, 2023.

[8] A. Wray and K. Ebrahimi, "Octovalve thermal management control for electric vehicles", *Energies*, 2022.

[9] R. Isermann and M. Münchhof, *Identification of Dynamic Systems: An Introduction with Applications*. Berlin, Germany: Springer, 2010.

[10] D. Grimm, "Model predictive control for the thermal system of an electric vehicle", Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2019.

[11] NTGD Valve, "Three-way valve", [Online]. Available: `https://ntgdvalve.com/three-way-valve/`. [Last accessed: Aug. 22, 2025].

[12] The Metal Company, "Differences between L-port and T-port valves", [Online]. Available: `https://www.themetalcompany.co.nz/faqs/differences-between-l-port-and-t-port-valves/?srsltid=AfmBOoq9QDOzE7iSM1F79wDkkqz3DVVcdbyiQkve_gMQ5WVUugHK5usa`. [Last accessed: Aug. 22, 2025].

[13] Michael Smith Engineers, "Centrifugal pumps" [Online]. Available: `https://www.michael-smith-engineers.co.uk/resources/useful-info/centrifugal-pumps#:~:text=A%20centrifugal%20pump%20is%20a,through%20the%20impeller's%20vane%20tips.`. [Last accessed: Aug. 22, 2025].

[14] A. Anzer, "Electrical drive system for electric vehicles (EVs)", *LinkedIn*, [Online]. Available: https://www.linkedin.com/pulse/electrical-drive-system-electric-vehicles-evs-anzer-usa/. [Last accessed: Aug. 22, 2025].

[15] TGlobal Corp., "Electric cars radiators", [Online]. Available: https://www.tglobalcorp.com/news/blog/electric-cars-radiators/. [Last accessed: Aug. 22, 2025].

[16] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ, USA: Prentice Hall, 1999.

[17] S. Santhakumaran, "Data-driven system identification and prediction of nonlinear processes", Doctoral thesis, Technische Universität Ilmenau, Ilmenau, Germany, 2023.

[18] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems", *Proc. Nat. Acad. Sci.*, 2016.

[19] U. Fasel, E. Kaiser, J. N. Kutz, B. W. Brunton, and S. Brunton, "SINDy with control: A tutorial", 2021.

[20] J. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control", *Mathematical Programming Computation*, 2018.

[21] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[22] N. Murgovski, *Model Predictive Control: Lecture Notes*. Chalmers University of Technology, 2024.

[23] ABB: Model predictive control technology demystified [Online]. Available: https://new.abb.com/industrial-software/features/model-predictive-control-mpc [Last accessed: Aug. 25, 2025].

[24] T. A. Johansen, "Introduction to nonlinear model predictive control and moving horizon estimation" [Online]. Available: https://api.semanticscholar.org/CorpusID:14712069, 2011. [Last accessed: Aug. 25, 2025].

[25] S. Gros and M. Diehl, "Numerical optimal control", Draft manuscript, 2022.

[26] ABB, "Model predictive control (MPC)", [Online]. Available: https://new.abb.com/industrial-software/features/model-predictive-control-mpc. [Last accessed: Aug. 22, 2025].

[27] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.

[28] Statology, "Z-score normalization", [Online]. Available: https://www.statology.org/z-score-normalization/. [Last accessed: Aug. 22, 2025].

[29] ResearchGate, "Sparse identification of nonlinear dynamics (SINDy)" [Online]. Available: https://www.researchgate.net/publication/301627530_Sparse_Identification_of_Nonlinear_Dynamics_SINDy. [Last accessed: Aug. 22, 2025].

[30] PySINDy Documentation, "Differentiation example", [Online]. Available: https://pysindy.readthedocs.io/en/stable/examples/5_differentiation/example.html. [Last accessed: Aug. 22, 2025].

[31] ScienceDirect, "Root-mean-square error", [Online]. Available: https://www.sciencedirect.com/topics/engineering/root-mean-square-error. [Last accessed: Aug. 22, 2025].

[32] M. Bayraktar, "What is SMAPE?", *Medium*, [Online]. Available: https://medium.com/@mertbayraktarxd/what-is-smape-2cf605831feb. [Last ccessed: Aug. 22, 2025].

[33] CasADi, "Direct multiple shooting (MATLAB example)", *GitHub*, [Online]. Available: https://github.com/casadi/casadi/blob/main/docs/examples/matlab/direct_multiple_shooting.m. [Last accessed: Aug. 22, 2025].

[34] COIN-OR Ipopt, "Ipopt Output," COIN-OR Documentation, [Online]. Available: https://coin-or.github.io/Ipopt/OUTPUT.html. [Last accessed: Aug. 26, 2025].

# A

# Appendix - Code

## A.1 SINDy, STLSQ implementation[29]

```
function Xi = sparsifyDynamics(Theta,dXdt,lambda,n)
% Copyright 2015, All Rights Reserved
% Code by Steven L. Brunton
% For Paper, "Discovering Governing Equations from Data:
%  Sparse Identification of Nonlinear Dynamical Systems"
% by S. L. Brunton, J. L. Proctor, and J. N. Kutz

% compute Sparse regression: sequential least squares
Xi = Theta\dXdt;  % initial guess: Least-squares

% lambda is our sparsification knob.
for k=1:10
    smallinds = (abs(Xi)<lambda);   % find small coefficients
    Xi(smallinds)=0;                % and threshold
    for ind = 1:n                   % n is state dimension
        biginds = ~smallinds(:,ind);
        % Regress dynamics onto remaining terms
        %to find sparse Xi
        Xi(biginds,ind) = Theta(:,biginds)\dXdt(:,ind);
    end
end
```

## A.2 DMS code, modified from [33]

```
% Copyright (C) 2010-2023 Joel Andersson, Joris Gillis,
% Moritz Diehl, KU Leuven.
% An implementation of direct multiple shooting

import casadi.*

T = 10; % Time horizon
N = 20; % number of control intervals

n = 0; % number of states
m = 0; % number of inputs

% Declare model variables
% In this case, n states and m inputs
x1 = SX.sym('x1');
...
x = [x1; ... ; xn];
u1 = SX.sym('u1');
...
u = [x1; ...; um];

% Model equations
xdot = zeros(n,1);
xdot(1) = ...;
...
xdot(n) = ...;

% Objective term
L = 0;

% Continuous time dynamics
f = Function('f', {x, u}, {xdot, L});

% Formulate discrete time dynamics
% Fixed step Runge-Kutta 4 integrator
M = 1; % RK4 steps per interval
DT = T/N/M;
f = Function('f', {x, u}, {xdot, L});
X0 = MX.sym('X0', n);
U = MX.sym('U',m);
X = X0;
Q = 0;
    for j=1:M
        [k1, k1_q] = f(X, U);
```

```
        [k2, k2_q] = f(X + DT/2 * k1, U);
        [k3, k3_q] = f(X + DT/2 * k2, U);
        [k4, k4_q] = f(X + DT * k3, U);
        X=X+DT/6*(k1 +2*k2 +2*k3 +k4);
        Q = Q + DT/6*(k1_q + 2*k2_q + 2*k3_q + k4_q);
    end
F = Function('F', {X0, U}, {X, Q}, {'x0','p'}, {'xf', 'qf'});


% Start with an empty NLP
w={};
w0 = [];
lbw = [];
ubw = [];
J = 0;
g={};
lbg = [];
ubg = [];

% "Lift" initial conditions
Xk = MX.sym('X0', n);
w = {w{:}, Xk};

%Define vectors for lower and upper bounds on w
lb_w = [];
ub_w = [];

%Define vector for initial condition
init = [];

lbw = [lbw; lb_w];
ubw = [ubw; ub_w];
w0 = [w0; init];

% Formulate the NLP
for k=0:N-1
    % New NLP variable for the control
    Uk = MX.sym(['U_' num2str(k)]);
    w = {w{:}, Uk};
    % Define vector for bounds on input
    lb_u = [];
    ub_u = [];
    % Define vector for initial input guess
    init_u = [];
    lbw = [lbw; lb_u];
    ubw = [ubw; ub_u];
```

```
    w0 = [w0;  init_u];

    % Integrate till the end of the interval
    Fk = F('x0', Xk, 'p', Uk);
    Xk_end = Fk.xf;
    J=J+Fk.qf;

    % New NLP variable for state at end of interval
    Xk = MX.sym(['X_' num2str(k+1)], n);
    w = [w, {Xk}];
    lbw = [lbw; lb_w];
    ubw = [ubw;  ub_w];
    w0 = [w0; init];

    % Add equality constraint
    g = [g, {Xk_end-Xk}];
    lbg = [lbg; 0; 0];
    ubg = [ubg; 0; 0];
end

% Create an NLP solver
prob = struct('f', J, 'x', vertcat(w{:}), 'g', vertcat(g{:}));
solver = nlpsol('solver', 'ipopt', prob);

% Solve the NLP
sol = solver('x0', w0, 'lbx', lbw, 'ubx', ubw,...
             'lbg', lbg, 'ubg', ubg);
w_opt = full(sol.x);
```

# B

# Appendix - Tables

## B.1 Model Performance Evaluation

**Table B.1:** Evaluation metrics for model with 18 states on training data

| Variable | Case 1 RMSE $\lambda = 0.01$ | Case 1 SMAPE[%] $\lambda = 0.01$ | Case 2 RMSE $\lambda = 0.5$ | Case 2 SMAPE[%] $\lambda = 0.5$ | Case 3 RMSE $\lambda = 0.07$ | Case 3 SMAPE[%] $\lambda = 0.07$ |
|---|---|---|---|---|---|---|
| $T_1$ | 2,99 | 19,83 | 0,32 | 2,39 | 0,17 | 1,19 |
| $T_2$ | 0,95 | 13,17 | 0,50 | 5,31 | 0,42 | 5,17 |
| $T_3$ | 1,02 | 12,16 | 0,42 | 4,58 | 0,24 | 2,81 |
| $T_4$ | 5,69 | 32,75 | 2,90 | 14,67 | 3,08 | 16,18 |
| $T_5$ | 0,46 | 2,51 | 0,13 | 0,75 | 0,10 | 0,57 |
| $T_6$ | 2,46 | 16,83 | 0,29 | 1,68 | 0,21 | 1,22 |
| $T_7$ | 0,64 | 4,90 | 0,26 | 2,15 | 0,26 | 2,17 |
| $T_8$ | 0,43 | 2,66 | 0,36 | 2,55 | 0,16 | 1,11 |
| $m_1$ | 0,42 | 4,41 | 0,07 | 0,66 | 0,05 | 0,51 |
| $m_2$ | 0 | 74,42 | 0,01 | 199,72 | 0,02 | 199,74 |
| $m_3$ | 0,42 | 6,90 | 0,07 | 1,13 | 0,04 | 0,71 |
| $m_4$ | 0,13 | 3,80 | 0,33 | 13,85 | 0,45 | 16,52 |
| $m_5$ | 59 | 198,93 | 38,48 | 198,79 | 19,08 | 196,68 |
| $m_6$ | 58,98 | 181,27 | 38,47 | 173,02 | 19,03 | 139,12 |
| $m_7$ | 0,38 | 3,86 | 0,04 | 0,68 | 0,04 | 1,03 |
| $m_8$ | 0 | 174,86 | 0,05 | 199,32 | 0,01 | 194,76 |
| $P_{bat}$ | 707,98 | 184,75 | 478,65 | 170,38 | 427,76 | 168,92 |
| $P_{ED}$ | 1,08 | 1,08 | 0,31 | 0,50 | 0,28 | 0,34 |

**Table B.2:** Evaluation metrics for model with 18 states on test data

| Variable | Case 1 RMSE $\lambda = 0.01$ | Case 1 SMAPE[%] $\lambda = 0.01$ | Case 2 RMSE $\lambda = 0.5$ | Case 2 SMAPE[%] $\lambda = 0.5$ | Case 3 RMSE $\lambda = 0.07$ | Case 3 SMAPE[%] $\lambda = 0.07$ |
|---|---|---|---|---|---|---|
| $T_1$ | 2,51 | 15,85 | 0,35 | 2,59 | 0,10 | 0,71 |
| $T_2$ | 1,34 | 16,80 | 0,46 | 5,60 | 0,50 | 5,91 |
| $T_3$ | 1,33 | 16,28 | 0,42 | 4,22 | 0,25 | 3,73 |
| $T_4$ | 9,53 | 44,75 | 3,43 | 17,28 | 3,56 | 17,64 |
| $T_5$ | 0,41 | 2,29 | 0,15 | 0,90 | 0,11 | 0,57 |
| $T_6$ | 6,07 | 73,51 | 0,32 | 1,87 | 0,24 | 1,48 |
| $T_7$ | 0,637 | 5,49 | 0,40 | 3,46 | 0,33 | 2,89 |
| $T_8$ | 0,53 | 3,01 | 0,36 | 2,20 | 0,19 | 1,26 |
| $m_1$ | 0,39 | 6,90 | 0,07 | 1,02 | 0,05 | 0,84 |
| $m_2$ | 0 | 88,01 | 0,02 | 199,73 | 0,02 | 199,74 |
| $m_3$ | 0,34 | 9,87 | 0,07 | 2,18 | 0,05 | 1,79 |
| $m_4$ | 0,17 | 5,12 | 0,48 | 19,60 | 0,48 | 22,42 |
| $m_5$ | 103,37 | 199,31 | 66,10 | 198,40 | 23,99 | 195,95 |
| $m_6$ | 103,08 | 189,93 | 66,56 | 175,45 | 24,24 | 147,54 |
| $m_7$ | 0,45 | 15,39 | 0,04 | 1,49 | 0,05 | 1,80 |
| $m_8$ | 0 | 185,91 | 0,08 | 199,56 | 0,01 | 195,09 |
| $P_{bat}$ | 971,24 | 184,25 | 697,17 | 172,63 | 644,05 | 180,76 |
| $P_{ED}$ | 1,04 | 2,14 | 0,28 | 0,71 | 0,29 | 0,90 |

**Table B.3:** Evaluation metrics for model with 9 states on training data

| Variable | Case 1 RMSE $\lambda = 0.1$ | Case 1 SMAPE[%] $\lambda = 0.1$ | Case 2 RMSE $\lambda = 0.01$ | Case 2 SMAPE[%] $\lambda = 0.01$ | Case 3 RMSE $\lambda = 0.001$ | Case 3 SMAPE[%] $\lambda = 0.001$ |
|---|---|---|---|---|---|---|
| $T_3$ | 0,63 | 5,82 | 0,19 | 1,73 | 0,13 | 1,27 |
| $T_4$ | 1,07 | 5,62 | 1,08 | 5,29 | 0,82 | 4,43 |
| $T_6$ | 0,50 | 2,81 | 0,94 | 5,27 | 0,26 | 1,61 |
| $T_8$ | 0,73 | 5,02 | 0,36 | 2,43 | 0,30 | 2,12 |
| $m_3$ | 0,22 | 6,70 | 0,12 | 3,58 | 0,15 | 4,15 |
| $m_5$ | 0 | 9,74 | 0,01 | 11,69 | 0 | 8,21 |
| $m_6$ | 0,6 | 11,53 | 1,08 | 11,02 | 0,54 | 9,52 |
| $m_7$ | 0,23 | 4,70 | 0,06 | 0,94 | 0,06 | 0,98 |
| $m_8$ | 0 | 169,13 | 0,01 | 196,86 | 0,13 | 198,78 |

**Table B.4:** Evaluation metrics for model with 9 states on test data

| Variable | Case 1 RMSE $\lambda = 0.1$ | Case 1 SMAPE[%] $\lambda = 0.1$ | Case 2 RMSE $\lambda = 0.01$ | Case 2 SMAPE[%] $\lambda = 0.01$ | Case 3 RMSE $\lambda = 0.001$ | Case 3 SMAPE[%] $\lambda = 0.001$ |
|---|---|---|---|---|---|---|
| $T_3$ | 0,66 | 6,51 | 0,19 | 1,64 | 0,13 | 1,11 |
| $T_4$ | 1,12 | 6,34 | 1,06 | 5,55 | 0,79 | 3,76 |
| $T_6$ | 0,51 | 2,90 | 0,99 | 5,29 | 0,34 | 1,83 |
| $T_8$ | 0,89 | 6,77 | 0,40 | 2,67 | 0,30 | 2,06 |
| $m_3$ | 0,23 | 5,53 | 0,14 | 2,85 | 0,25 | 3,87 |
| $m_5$ | 0 | 6,82 | 0,01 | 11,70 | 0 | 7,40 |
| $m_6$ | 0,61 | 11,30 | 0,93 | 10,02 | 0,55 | 9,24 |
| $m_7$ | 0,26 | 4,82 | 0,06 | 1,010 | 0,06 | 1,65 |
| $m_8$ | 0 | 163,26 | 0,02 | 196,71 | 0,13 | 195,16 |

## B.2 Model Comparison Evaluation

**Table B.5:** Evaluation metrics for linear model on training data (unscaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 1e^{-4}$ | Case 1 SMAPE[%] $\lambda = 1e^{-4}$ | Case 2 RMSE $\lambda = 1e^{-6}$ | Case 2 SMAPE[%] $\lambda = 1e^{-6}$ | Case 3 RMSE $\lambda = 1e^{-5}$ | Case 3 SMAPE[%] $\lambda = 1e^{-5}$ |
|---|---|---|---|---|---|---|
| $T_3$ | 0,89 | 8,75 | 0,96 | 8,91 | 0,89 | 8,73 |
| $T_4$ | 0,48 | 2,22 | 0,44 | 2,29 | 0,48 | 2,22 |
| $T_6$ | 0,52 | 3,08 | 0,51 | 2,93 | 0,52 | 3,08 |
| $T_8$ | 0,22 | 1,36 | 0,39 | 2,67 | 0,22 | 1,35 |
| $m_3$ | 0,39 | 6,32 | 0,21 | 4,90 | 0,39 | 6,28 |
| $m_5$ | 0,07 | 75,70 | 0,09 | 75,24 | 0,07 | 80,52 |
| $m_6$ | 0,52 | 9,04 | 0,24 | 5,10 | 0,52 | 9 |
| $m_7$ | 0,07 | 1,07 | 0,39 | 4,24 | 0,07 | 1,05 |
| $m_8$ | 0,32 | 199,65 | 0 | 167,04 | 0,33 | 199,65 |

**Table B.6:** Evaluation metrics for linear model on test data (unscaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 1e^{-4}$ | Case 1 SMAPE[%] $\lambda = 1e^{-4}$ | Case 2 RMSE $\lambda = 1e^{-6}$ | Case 2 SMAPE[%] $\lambda = 1e^{-6}$ | Case 3 RMSE $\lambda = 1e^{-5}$ | Case 3 SMAPE[%] $\lambda = 1e^{-5}$ |
|---|---|---|---|---|---|---|
| $T_3$ | 0,93 | 8,85 | 0,92 | 9,07 | 0,92 | 9,03 |
| $T_4$ | 0,47 | 2,32 | 0,44 | 2,05 | 0,44 | 2,05 |
| $T_6$ | 0,47 | 2,70 | 0,51 | 2,95 | 0,51 | 2,95 |
| $T_8$ | 0,38 | 2,62 | 0,21 | 1,30 | 0,22 | 1,29 |
| $m_3$ | 0,30 | 3,85 | 0,36 | 5,43 | 0,37 | 5,51 |
| $m_5$ | 0,09 | 70,94 | 0,08 | 74,26 | 0,09 | 82,42 |
| $m_6$ | 0,23 | 4,05 | 0,52 | 7,26 | 0,53 | 7,35 |
| $m_7$ | 0,40 | 4,74 | 0,07 | 1,02 | 0,07 | 0,96 |
| $m_8$ | 0 | 159,87 | 0,37 | 199,63 | 0,37 | 199,54 |

**Table B.7:** Evaluation metrics for polynomial model of order 2 on training data (unscaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 1$ | Case 1 SMAPE[%] $\lambda = 1$ | Case 2 RMSE $\lambda = 0.01$ | Case 2 SMAPE[%] $\lambda = 0.01$ | Case 3 RMSE $\lambda = 1e^{-4}$ | Case 3 SMAPE[%] $\lambda = 1e^{-4}$ |
|---|---|---|---|---|---|---|
| $T_3$ | 0,40 | 4,18 | 0,19 | 1,73 | 0,15 | 1,67 |
| $T_4$ | 0,64 | 3,53 | 1,09 | 5,30 | 0,97 | 4,64 |
| $T_6$ | 0,50 | 2,80 | 0,94 | 5,27 | 0,43 | 2,94 |
| $T_8$ | 0,70 | 4,55 | 0,36 | 2,43 | 0,17 | 1,16 |
| $m_3$ | 0,31 | 6,71 | 0,12 | 3,58 | 0,29 | 5,25 |
| $m_5$ | 0 | 9,54 | 0,01 | 11,74 | 0,06 | 76,34 |
| $m_6$ | 0,60 | 11,50 | 1,06 | 11,50 | 1,01 | 16,95 |
| $m_7$ | 0,36 | 7,18 | 0,06 | 0,94 | 0,06 | 1,837 |
| $m_8$ | 0 | 181,73 | 0,01 | 196,87 | 0,06 | 199,06 |

**Table B.8:** Evaluation metrics for polynomial model of order 2 on test data (unscaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 1$ | Case 1 SMAPE[%] $\lambda = 1$ | Case 2 RMSE $\lambda = 0.01$ | Case 2 SMAPE[%] $\lambda = 0.01$ | Case 3 RMSE $\lambda = 1e^{-4}$ | Case 3 SMAPE[%] $\lambda = 1e^{-4}$ |
|---|---|---|---|---|---|---|
| $T_3$ | 0,37 | 4,28 | 0,19 | 1,64 | 0,14 | 1.82 |
| $T_4$ | 0,57 | 2,96 | 1,06 | 5,54 | 1,11 | 5.86 |
| $T_6$ | 0,52 | 2,89 | 0,99 | 5,28 | 0,64 | 4.33 |
| $T_8$ | 0,93 | 7,12 | 0,40 | 2,67 | 0,16 | 1.12 |
| $m_3$ | 0,36 | 10,18 | 0,14 | 2,86 | 0,34 | 5.50 |
| $m_5$ | 0 | 8,82 | 0,01 | 11,66 | 0,08 | 88.19 |
| $m_6$ | 0,60 | 11,32 | 0,94 | 9,81 | 0,86 | 15.54 |
| $m_7$ | 0,39 | 7,59 | 0,06 | 1,01 | 0,06 | 2.13 |
| $m_8$ | 0 | 149,75 | 0,02 | 196,71 | 0,08 | 199.18 |

**Table B.9:** Evaluation metrics for linear model on training data (scaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 0.001$ | Case 1 SMAPE[%] $\lambda = 0.001$ | Case 2 RMSE $\lambda = 1e^{-4}$ | Case 2 SMAPE[%] $\lambda = 1e^{-4}$ |
|---|---|---|---|---|
| $T_3$ | 5,35 | 40,40 | 5,64 | 38,62 |
| $T_4$ | 0,62 | 3,48 | 0,49 | 2,47 |
| $T_6$ | 3,16 | 16,63 | 4,22 | 23,27 |
| $T_8$ | 2,10 | 13,75 | 2 | 13,22 |
| $m_3$ | 1,13 | 24,80 | 0,82 | 24,41 |
| $m_5$ | 0 | 10,37 | 0,80 | 191,46 |
| $m_6$ | 0,19 | 2,55 | 0,94 | 18,56 |
| $m_7$ | 0,47 | 8,92 | 0,47 | 9,88 |
| $m_8$ | 0,99 | 199,71 | 0,59 | 199,75 |

**Table B.10:** Evaluation metrics for linear model on test data (scaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 0.001$ | Case 1 SMAPE[%] $\lambda = 0.001$ | Case 2 RMSE $\lambda = 1e^{-4}$ | Case 2 SMAPE[%] $\lambda = 1e^{-4}$ |
|---|---|---|---|---|
| $T_3$ | 6,86 | 47,25 | 6,82 | 45,05 |
| $T_4$ | 0,62 | 3,67 | 0,46 | 2,30 |
| $T_6$ | 3,70 | 16,64 | 4,23 | 23,39 |
| $T_8$ | 2,18 | 13,52 | 2,19 | 13,05 |
| $m_3$ | 0,92 | 15,97 | 0,87 | 23,71 |
| $m_5$ | 0 | 5,91 | 0,83 | 191,38 |
| $m_6$ | 0,16 | 2 | 0,90 | 17,50 |
| $m_7$ | 0,47 | 9,09 | 0,45 | 11,98 |
| $m_8$ | 0,64 | 184,10 | 0,72 | 199,73 |

**Table B.11:** Evaluation metrics for polynomial model of order 2 on training data (scaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 1$ | Case 1 SMAPE[%] $\lambda = 1$ | Case 2 RMSE $\lambda = 0.1$ | Case 2 SMAPE[%] $\lambda = 0.1$ |
|---|---|---|---|---|
| $T_3$ | 7,68 | 55 | 10,29 | 90,50 |
| $T_4$ | 0,64 | 3,59 | 0,56 | 3,07 |
| $T_6$ | 0,58 | 3,42 | 2,62 | 17,13 |
| $T_8$ | 1,92 | 12,30 | 1,92 | 12,30 |
| $m_3$ | 0,85 | 28,71 | 0,40 | 24,37 |
| $m_5$ | 0,15 | 130,22 | 0,04 | 50,26 |
| $m_6$ | 0,47 | 13 | 0,35 | 9,53 |
| $m_7$ | 0,37 | 8,55 | 0,47 | 9,58 |
| $m_8$ | 0,01 | 197,59 | 0,15 | 199,47 |

**Table B.12:** Evaluation metrics for polynomial model of order 2 on test data (scaled pump inputs)

| Variable | Case 1 RMSE $\lambda = 1$ | Case 1 SMAPE[%] $\lambda = 1$ | Case 2 RMSE $\lambda = 0.1$ | Case 2 SMAPE[%] $\lambda = 0.1$ |
|---|---|---|---|---|
| $T_3$ | 7,51 | 56,55 | 14,02 | 90,09 |
| $T_4$ | 0,64 | 3,66 | 0,52 | 2,76 |
| $T_6$ | 0,64 | 3,56 | 2,88 | 18,55 |
| $T_8$ | 2,06 | 12,63 | 2,06 | 12,63 |
| $m_3$ | 0,95 | 26,62 | 0,43 | 26,66 |
| $m_5$ | 0,21 | 139,27 | 0,05 | 49,26 |
| $m_6$ | 0,47 | 12,16 | 0,38 | 9,07 |
| $m_7$ | 0,45 | 8,27 | 0,44 | 10,89 |
| $m_8$ | 0,01 | 196,62 | 0,17 | 199,43 |

**DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS
UNIVERSITY OF TECHNOLOGY