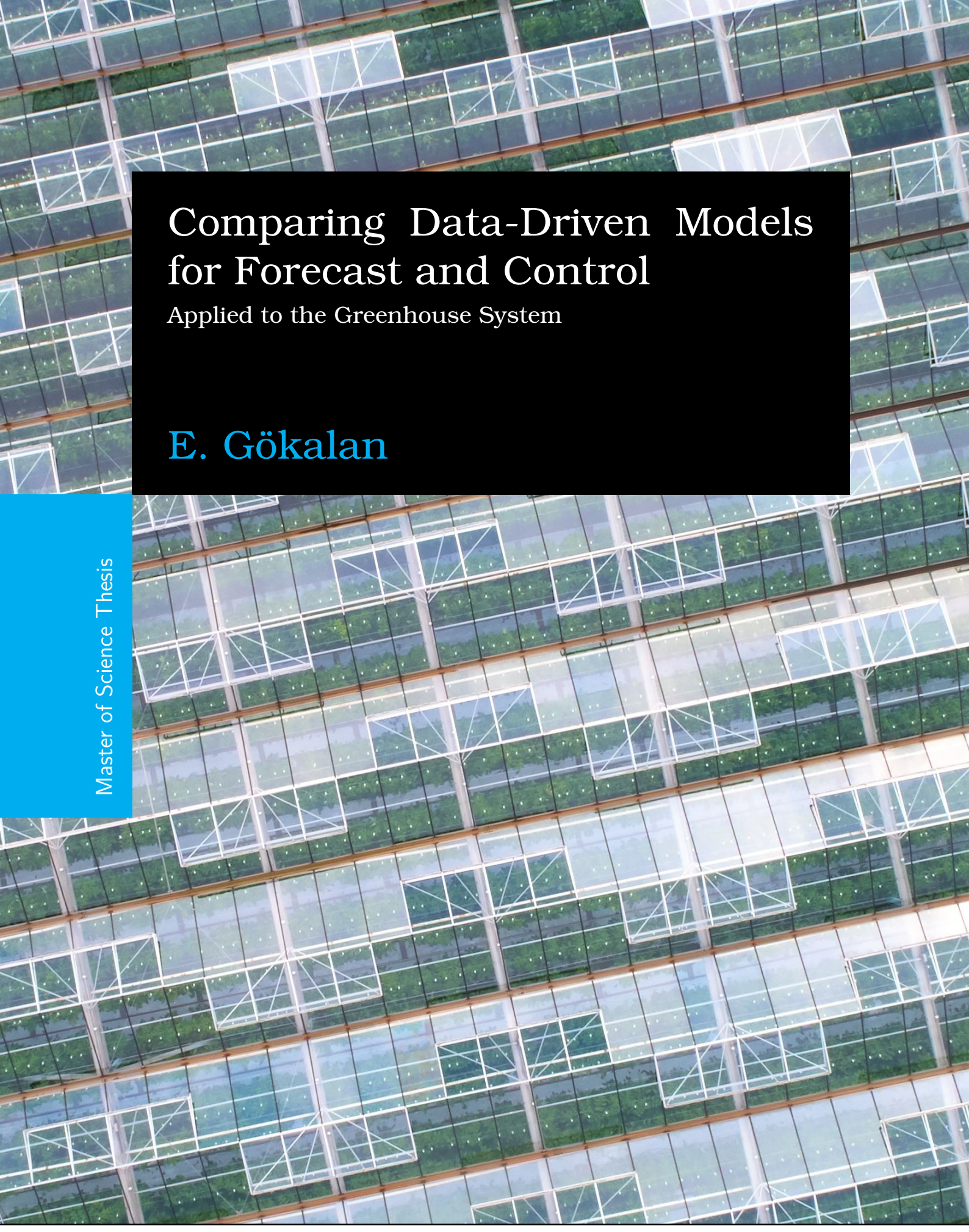

Comparing Data-Driven Models for Forecast and Control

Applied to the Greenhouse System

E. Gökalan

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Comparing Data-Driven Models for Forecast and Control

Applied to the Greenhouse System

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

E. Gökalan

January 4, 2024

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Greenhouses offer the promise to mitigate the challenges faced by traditional open field agriculture. Operating these systems on a commercial scale demands effective control and forecast models. This thesis contributes to the increasing field of research that applies methods from systems and control to greenhouse systems. The primary objective was to evaluate various climate prediction models for their applicability in forecast and predictive control within greenhouses. Comparative analyses were conducted on linear and non-linear data-driven models across one week and one hour time horizons. The linear models include an autoregressive with exogenous input (ARX) model, subspace identification using the Multivariable Output-Error State-space algorithm (MOESP), and dynamic mode decomposition with control (DMDc). Of the linear methods, MOESP performed the best on both time horizons. The nonlinear methods included several long-short term memory (LSTM) models with different architectures. A stacked LSTM model was found to outperform the other LSTM models as well as all linear models over the one-week time horizon. The Koopman with inputs and control (KIC) framework, a novel nonlinear approach, was also examined. The choice of lifting functions proved to be a challenge, yielding only partial improvements over linear methods on the training set. For the short-term one-hour time horizon, the data-enabled predictive control (DeePC) algorithm was also implemented and yielded the most accurate results. Notably, the stacked LSTM predictor did not show any advantages on the one-hour timescale. Finally, a novel approach combining Koopman theory with DeePC was introduced and compared to the regular DeePC algorithm. No improvement was observed with this proposed method for one hour-ahead predictions. The comparison concluded with the insight that for the one hour time-horizon, the nonlinear methods that were investigated did not show advantages over linear methods in the greenhouse system. On the one week time horizon, nonlinear methods outperform linear ones. It affirmed the challenges around finding observables that fully span a Koopman invariant subspace. Further investigation on the proposed extension to DeePC should be the focus of future work.

Preface

Over the course of my academic career, the climate and ecological crisis has become the central motivation for my work. As I continued to witness a global failure to address this crisis with the urgency and vigor that it demands, the threat of increasing food insecurity became my motivation for applying systems and control principles to the domain of agriculture.

While this drive to address the crisis also resulted in a lot of time spent in the political arena at the university and beyond, I have to wholeheartedly thank my supervisor Prof.dr.ir.Keviczky for refocusing me on my academic work. Without his firm guidance and continued support, I would not have been able to complete my work. Many thanks also have to be extended to Ioannis Panagopoulos. His help and support when I struggled with applications of Koopman theory was invaluable.

I am forever grateful for the love, support and patience of my family. Their love has been a corner stone I could always rely on and they helped me develop into the person I am today. Lastly, I have to thank my many friends that have been by my side throughout this process. To all those who gave me mental support during stressful exam periods and the final stretches of the thesis, you know who you are and my thanks go out to you. Special thanks go out to Lars, who's honest words always helped me reflect and to Anne-Linn, who's wraps kept me fed during long days in the library.

Delft, University of Technology

January 4th, 2023

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1-1 Motivation

As the climate crisis continues to threaten food security around the globe by reducing arable land [1], [7], controlled environment agriculture is one of the adaptation technologies to guarantee yields in a changing climate.

Controlled environment agriculture offers a solution to adapt to a changing climate since it uses less water and requires less land for the same output as traditional open field agriculture [2]. Glass greenhouses are one of these controlled agriculture systems and are very successfully deployed in the Netherlands, where the majority of agricultural land is under a greenhouse [38].

One of the main challenges of the greenhouse sector today is energy consumption. In the Netherlands, greenhouses account for 5% of the total energy consumption of the country [11].

As a consequence, energy consumption plays a major role in the economics of a greenhouse. Through the use of forecast models, the energy consumption of a greenhouses can be reduced [33]. These forecast models rely on weather predictions and an underlying greenhouse climate model to guide the decision-making of the growers to reduce energy consumption and ensuring profitable yields.

Accurate temperature predictions within greenhouses are also important as the temperatures impact crop growth significantly [42]. Maintaining suitable temperatures is crucial for enhancing plant quality, quantity, and mortality rates. The use of model predictive control algorithms has become popular in the search to find more optimal ways to control the greenhouse temperature[16].

Thus, there exists a demand for accurate climate predictions, both for model predictive control applications, as well as for economic forecast models. Many different approaches can be taken to mathematically model a greenhouse system, but according to Yu et al. (2016) [40] there are two main approaches. The first type is the physics-based model, which tries to model the underlying thermodynamic mechanisms that affect the change of the climate. They are

typically made up of differential equations that are derived from thermodynamic principles to describe the climate. These models typically have a large number of parameters. The second type of model are data-driven models. As it is often difficult to measure every parameter for the physics-based models, data-driven models aim to determine the system behaviour by collecting measurement data.

The focus of this research is a comparison of the second model type for forecast and predictive control applications. Furthermore, the Koopman with inputs and control (KIC) framework by Proctor et al. (2018) [29] has gained a lot of attention in recent years as a novel, data-driven modelling technique.

The Koopman operator was first developed by Bernard Koopman (1931) [20] and only started to get attention in recent years, initially to investigate spectral properties (Mezic et al. [25]) and later to be applied for some engineering-oriented applications (Budivsic et al.) [5]. This method has already seen applications in combination with model predictive control (MPC) and shows promising potential to control multi-dimensional nonlinear systems [3]. This modelling approach has not been applied to the greenhouse system yet.

1-2 Contribution

The greenhouse is a nonlinear system where in some cases, a lot of sensor data is already being collected. There has been an increasing interest in Data-driven system identification and control approaches for nonlinear systems in a world that increasingly generates more data.

This thesis compares some existing data-driven climate modelling approaches in two distinct applications for the greenhouse system. Firstly, it compares the models on their performance as forecast models for the time horizon of one week. Secondly, it compares the modelling approaches on their performance for the application in model predictive control, where the time horizon is one hour.

The performance of these models is compared on the two different time horizons. Two main modelling paradigms are compared. The first is models that rely on an underlying linear structure to create forecasts, and the second is models that contain nonentities within their structure. Lastly, models that have been designed specifically with the application of model predictive control have been included in the comparison focused on predictive control performance.

The linear models included are an auto-regressive with exogenous input (ARX) model, a subspace identification model using the MOESP algorithm and a model based on the dynamic mode decomposition with control. Furthermore, in the evaluation of predictive performance for control, the Data enable predictive control (DeePC) algorithm is included in the comparison.

The nonlinear models that are compared are firstly, LSTM implementations with different architectures and secondly, a Koopman based predictor that uses the KIC framework that was first proposed in [29].

LSTMs have been applied to greenhouses before [17], [23],[34]. While these implementations typically use single-layer LSTM architectures, a stacked LSTM architecture is also implemented and evaluated.

The Koopman based approach has not been applied to the greenhouse system yet. Its implementation is evaluated and the problems that arose from the implementation are analysed and described.

Lastly, an extension to the Data-Enabled Predictive Control (DeePC) approach is proposed and compared to the original algorithm. The extension seeks to combine the KIC framework with DeePC in order to improve predictions over the time horizon of one hour.

There are several contributions of this work. The first is the implementation and comparison between single layer and stacked LSTM architectures for the greenhouse system. The second contribution is the implementation and evaluation of a Koopman based predictor for the greenhouse system, which was the original motivation for this thesis. Furthermore a proposed extension of the DeePC control algorithm is presented. Lastly, insight is gained into the performance of each modelling type on different time horizons, as well as recommendations for how to improve future data-driven greenhouse modelling approaches.

1-3 Research Question

While the original research question of this project was focused on the implementation of a controller based on a Koopman predictor, the focus shifted towards a comparison of different data-driven modelling approaches for different time horizons. This resulted in the following research question:

How do different data-driven modelling approaches compare as forecast and predictive control models when applied to greenhouse data?

Several sub-questions were derived to further investigate this question:

- How does the selection of linear models compare to the selection of nonlinear models for the purpose of week ahead forecast?
- How can a predictor using the Koopman in control (KIC) framework be implemented and what are its challenges?
- What data-driven modelling approaches included in the comparison are best suited for model predictive control in the greenhouse system?
- How can a Koopman theory be included in the formulation of the data enabled predictive control algorithm and how does it perform?

1-4 Outline

This work has the following structure: In Chapter 2, the ground truth model that provides the data for all subsequent modelling approaches is described. A sub-system of this ground truth model is chosen to be the focus of analysis and its governing equations are shown. Chapter 3 will discuss linear data-driven modelling methods and compare their performance as forecast

models. In Chapter 4, different implementations of a Long-Short-Term Memory (LSTM) model are presented and compared to the linear forecast models of chapter 3. In chapter 5, Koopman theory and the KIC predictor are explained and the implementation of the KIC predictor for the greenhouse system is discussed and analysed. In chapter 6, the predictive performance on the one-hour time horizon of the previously derived models is evaluated. Furthermore, the data-enabled predictive control (DeePC) algorithm is presented and its implementation is compared. Additionally, a proposed extension to the DeePC algorithm is included in the chapter and its implementation is also compared. Lastly, Chapter 7 will give a summary and answer the research questions that were set out in the beginning as well as give some recommendations for future work.

Chapter 2

The Ground Truth Model - GreenLight

In order to create data-driven greenhouse climate models, a data source is required. The following chapter will discuss the GreenLight model by Katzin et al. [19] as it was chosen to form the ground truth model for the subsequent data-driven modelling approaches. The GreenLight model was chosen due to its open source nature as well as having been experimentally validated. First, the full GreenLight model is discussed. After that, a selection of states that will be approximated in the data driven approaches in the following chapters is described. Lasly, the underlying equations that GreenLight uses are shown.

2-1 The GreenLight Model

The GreenLight model is a physics based model with many parameters and is based on the already validated model by Vanthoor (2011) [35]. Furthermore, the GreenLight model itself was validated using measured experimental data. It was shown that the model simulated temperatures accurately, with some deviations in CO_2-levels and relative humidity. In the next subsections, the states that GreenLight considers are discussed to give an understanding of the granularity of the model.

2-1-1 States

Temperature

The temperature is affected by solar radiation and the outside temperature, ventilation, wind as well as any heating, cooling or fogging system inside the greenhouse. Furthermore, the plant itself will also interact with the temperature mainly through evaporating water.

In the GreenLight model, the temperature state is described by 17 different state variables. These include five different soil temperatures T_{So1} ... T_{So5} and four states for the lamp temperature T_{Lamp}, $T_{IntLamp}$, $T_{GroPipe}$ and T_{BlScr}, which describe the temperature of the top

lights, the inter lights, the grow pipes and the blackout screens, respectively. Furthermore, there is a state for the temperature on the external side of a cover $T_{Cov,e}$, the internal side of a cover $T_{Cov,in}$, the air above the thermal cover T_{Top}, the thermal screen T_{ThScr}, the air in the main compartment T_{air}, the canopy T_{Can}, the pipe rail system T_{Pipe} and the floor T_{Flr}. It should be noted that the cover in this model represents a lumped cover that includes all elements of the permanent greenhouse cover. All temperature states are given Degree centigrade [C°].

$$\mathbf{T_{Sol}}(\mathbf{t}) = \begin{bmatrix} T_{Sol1}(t) \\ \vdots \\ T_{Sol5}(t) \end{bmatrix}, \qquad \mathbf{T_{Light}}(\mathbf{t}) = \begin{bmatrix} T_{Lamp}(t) \\ T_{IntLamp}(t) \\ T_{GroPipe}(t) \\ T_{BlScr}(t) \end{bmatrix} \tag{2-1}$$

$$\mathbf{T}(\mathbf{t}) = \begin{bmatrix} T_{Cov,e}(t) \\ T_{Cov,in}(t) \\ T_{ThScr}(t) \\ T_{air}(t) \\ T_{Can}(t) \\ T_{Pipe}(t) \\ T_{Flr}(t) \\ \mathbf{T_{Sol}}(\mathbf{t}) \\ \mathbf{T_{Light}}(\mathbf{t}) \end{bmatrix} \tag{2-2}$$

Humidity

In the GreenLight model, the vapor pressure is used as the measure for humidity. It is taken at two locations, in the main compartment and at the top. The states are denoted by VP_{Air} and VP_{Top}, respectively and use the pressure unit Pa. Equation 2-3 represents the humidity state of GreenLight:

$$\mathbf{VP}(\mathbf{t}) = \begin{bmatrix} VP_{Air}(t) \\ VP_{Top}(t) \end{bmatrix} \tag{2-3}$$

CO_2 concentration

The CO_2 concentration also modelled in two different locations in a similar manner as is done for the vapor pressure. Thus, the concentration is modelled in the main compartment and at the top and is denoted by CO_2Air and CO_2Top, respectively. The state vector for the CO_2 concentration is given by:

$$\mathbf{CO_2}(\mathbf{t}) = \begin{bmatrix} CO_2Air(t) \\ CO_2Top(t) \end{bmatrix} \tag{2-4}$$

The state vector for the climate of the greenhouse of the example model can be fully defined by the vectors for each relevant climate variable. This is summarised by expression 2-5, where the overall state vector is compromised by the concatenated climate variable vectors.

$$\mathbf{x(t)} = \begin{bmatrix} \mathbf{T(t)} \\ \mathbf{VP(t)} \\ \mathbf{CO_2(t)} \end{bmatrix} \tag{2-5}$$

2-1-2 Exogenous Signals

Disturbances

The GreenLight model considers seven different disturbances that act on the greenhouse system. These are outside global radiation $I_{Glob}(Wm^{-2})$, outside temperature T_{out} (C°), the temperature of the sky T_{sky} (C°), outer soil temperature T_{SoOut} (C°), outside vapour pressure VP_{Out} (Pa), outside CO_2 concentration CO_{2Out} (ppm) and wind speed v_{wind}. Together these external disturbances can be collected in a vector as shown in equation 2-6

$$\mathbf{d(t)} = \begin{bmatrix} I_{Glob}(t) \\ T_{out}(t) \\ T_{sky}(t) \\ T_{SoOut}(t) \\ VP_{Out}(t) \\ CO_{2Out}(t) \\ d_{wind}(t) \end{bmatrix} \tag{2-6}$$

Inputs

Multiple inputs are considered in the GreenLight model. They all take on values in the range [01], with 0 being no actuation and 1 being full actuation. The vector of inputs is given by:

$$\mathbf{in(t)} = \begin{bmatrix} Boil(t) \\ BoilGro(t) \\ ExtCo(t) \\ Roof(t) \\ ThScr(t) \\ BlScr(t) \\ Lamp(t) \\ IntLamp(t) \end{bmatrix} \tag{2-7}$$

All inputs are given on a continous range, with the exception of Lamp and IntLamp . These lamp inputs are actuated discretely.

2-1-3 Crop Model of GreenLight

The crop model in the GreenLight model is a simplified one when compared to the original Vanthoor (2011) [35] model. The GreenLight model employs a single stage crop model that only considers dry mass of the fruit. It does not consider multiple crop development stages. This was done to simplify simulation since the goal of this model was to simulate the effect of different lighting systems. It also does not consider number of fruits. Instead it considers the total fruit dry mass. The crop states are summarized by:

$$x_c = \begin{bmatrix} C_{Buf} \\ C_{Leaf} \\ C_{Stem} \\ C_{Fruit} \\ T_{Can_{Sum}} \\ T_{Can_{24}} \end{bmatrix} \tag{2-8}$$

The term C_{Buf} refers to the mass of accumulated carbon-hydrate buffer and C_{Leaf}, C_{Stem} and C_{Fruit} refer to the carbon-hydrates in the leaves, stem and fruit, respectively. The average temperature canopy rate $T_{Can_{24}}$ is related to the fruit growth period and the size of the fruit. The integral of the canopy temperature $T_{Can_{Sum}}$ experienced by the plant is used to indicate the development of the crop.

The focus of this work was set on predicting the climate states of the greenhouse over one week and one hour time horizons. On these time horizons, the crop development did not have a significant influence on the predictions. The energy balance of the greenhouse is the most relevant to predicting future climate states. The energy balance is determined through radiative, convective, conductive and latent heat exchanges. The radiative heat exchanges are broken up into photosynthetically active radiation (PAR), near infrared radiation (NIR) and far infrared radiation (FIR). This energy balance is summarised in figure 2-1. All heat exchanges in the simulation are in the unit of Wm^{-2} and the evolution of the heat exchanges is determined through differential equations.

2-2 The Sub-System in Consideration

For the purpose of comparing Data-Driven modelling approaches, a subsystem of the Green-Light model is considered. This has several reasons, but the main reason is that considering all states and exogenous input signals during for each method requires significantly more computing power and increases the complexity of the problem.

Hence, a subsystem was chosen. Since the temperature states are the most relevant for a crop and due to the fact that they were also the most accurately approximated states by the GreenLight model [19], a focus was set on temperature prediction.

The temperatures that were considered were the air temperature T_{Air} and canopy temperature T_{Can}. For the choice of exogenous inputs, they were chosen to be the most relevant for these two temperatures. The external disturbances that were taken into consideration were the global radiation I_{Glob}, the outside temperature T_{out} and the wind speed d_{wind}. The most

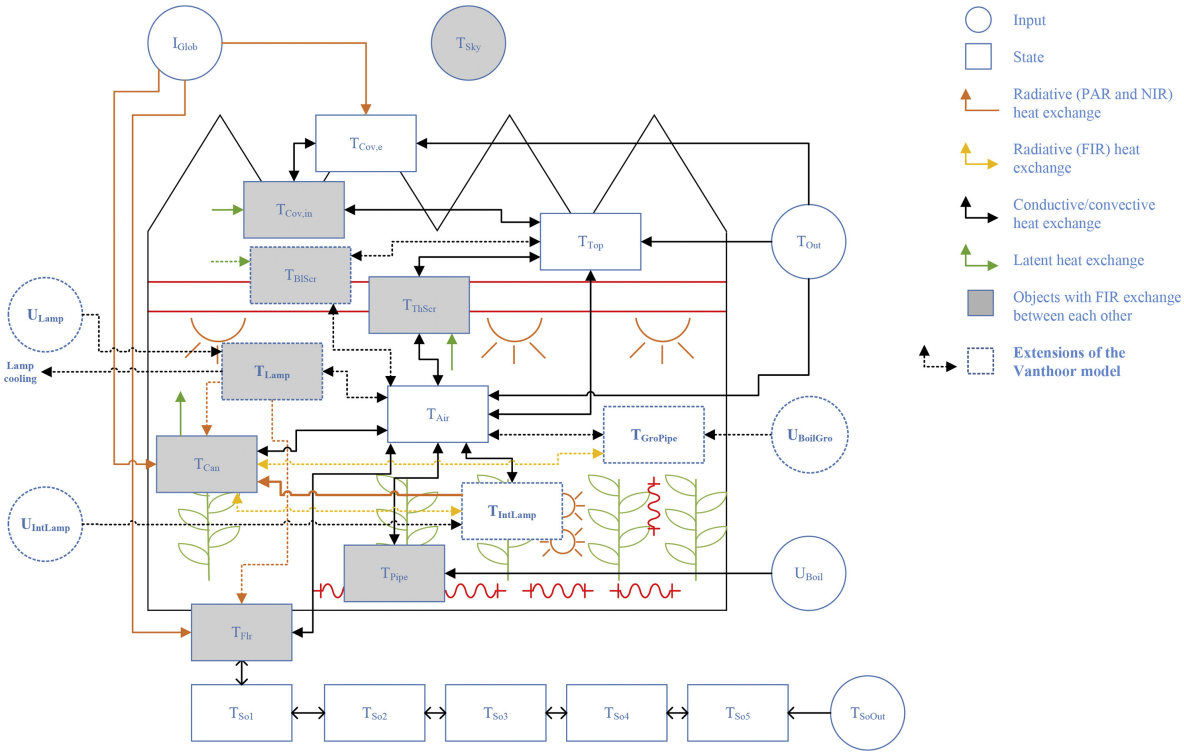

Figure 2-1: Schematic of the energy balance of the GreenLight model, where dashed and bold lines are additions to the Vanthoor model. All items in Grey exchange FIR between each other [19]

relevant control inputs were chosen to be the pipe temperature T_{Pipe}, the temperature of the grow pipe $T_{GroPipe}$, the thermal screen u_{thScr} and the roof u_{roof}. The input signals for the thermal screen u_{thScr} and the roof u_{roof} are values between zero and one that indicate how open or closed they are. A value of zero means it is fully closed and a value of one reflects it being fully open. The temperatures of the pipes, while not true control inputs, are closely linked to the state of the two boiler inputs u_{boil} and $u_{groBoil}$. It is assumed that the relation between these boiler inputs and the temperature of the pipes is known and hence the temperature of the pipe is used.

The goal with the prediction and forecast models is to find a system approximation that takes initial temperatures $y_0(k)$, known inputs and disturbances over some time horizon from $k+1$ to N, and predict temperatures over this time-horizon. Essentially capturing the relationship between the external signals and the temperature inside. An overview of the subsystem that is supposed to be approximated is given by figure 2-2

It is worth noting that the weather disturbances $d(k+1)$ to $d(N)$ are treated as known variables, while in reality these can only be approximated for any future time horizon N. Still, any prediction model for a greenhouse in practice has to rely on weather forecasts, since it significantly affects the temperature within the greenhouse. So for the purpose of week-long forecasts and for the purpose of short-term prediction for control, it is assumed that the weather disturbances are known or at least estimated. This results in the combined exogenous input vector for time-step k to be expressed by 2-9.

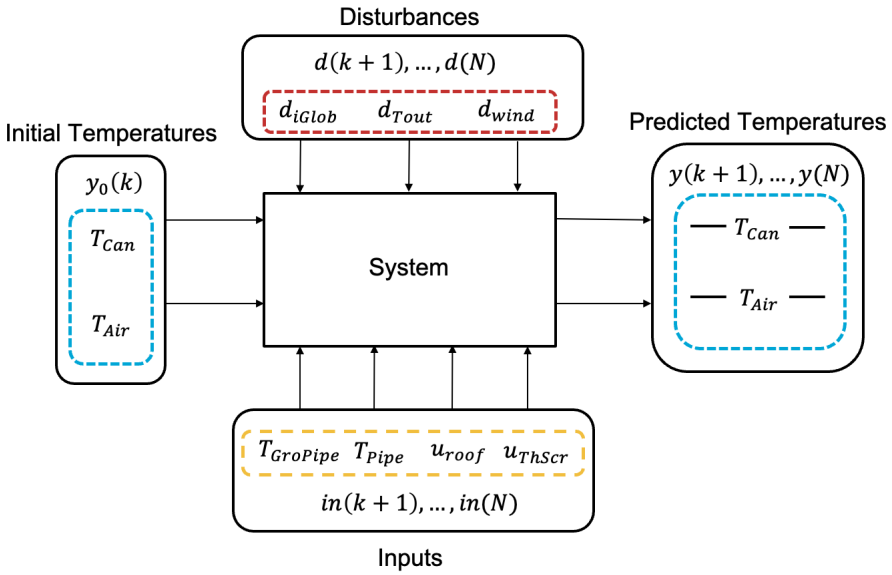

Figure 2-2: Chosen subsystem to be approximated by the data-driven modelling approaches

$$u(k) = \begin{bmatrix} in(k) \\ d(k) \end{bmatrix} \tag{2-9}$$

In all subsequent model estimations, this combined vector is used and is referred to as the input.

2-2-1 Governing Equations of the Subsystem

Within the simulation of GreenLight, these Temperatures are calculated through differential equations. The differential equation used for the evolution of the temperature T_{Air} is given by:

$$\begin{aligned} \text{cap}_{\text{Air}} \dot{T}_{\text{Air}} =& H_{\text{CanAir}} + H_{\text{PipeAir}} + R_{Glob\_SunAir} - H_{\text{AirFlr}} - H_{\text{AirThScr}} \\ & - H_{\text{AirOut}} - H_{\text{AirTop}} - H_{\text{AirBlScr}} + H_{\text{LampAir}} \\ & + R_{\text{LampAir}} + H_{\text{IntLampAir}} + H_{\text{GroPipeAir}} \end{aligned} \tag{2-10}$$

And for T_{Can} this differential equation is:

$$\begin{aligned} \text{cap}_{\text{Can}} \dot{T}_{\text{Can}} =& R_{\text{PARSunCan}} + R_{\text{NIRSunCan}} + R_{\text{PipeCan}} - H_{\text{CanAir}} - L_{\text{CanAir}} \\ & - R_{\text{CanCov, in}} - R_{\text{CanFlr}} - R_{\text{CanSky}} - R_{\text{CanThScr}} - R_{\text{CanBlScr}} \\ & + R_{\text{PARLampCan}} + R_{\text{NIRLampCan}} + R_{\text{FIRLampCan}} \\ & + R_{\text{PARIntLampCan}} + R_{\text{NIRIntLampCan}} + R_{\text{FIRIntLampCan}} \\ & + R_{\text{GroPipeCan}} \end{aligned} \tag{2-11}$$

In these equations, terms denoted by R are radiative heat exchanges, with FIR and PAR referring to the spectrum radiation of the heat exchanged. Terms denoted as H are convective

and L are latent heat exchanges and the subscript denotes the source of the heat exchange as well as its endpoint. Hence, the term R_{PipeCan} would be the radiative heat exchange from the pipe to the canopy and H_{CanAir} would be the convective heat exchange from the canopy to the air temperature. A summary of all relevant subscripts can be found in table 2-1.

Air	main greenhouse comparement below the thermal screen	Lamp	Top lights
Glob	Global radiation	InLamp	Inter lights
Can	Canopy	ThScr	Thermal screen
Flr	Floor	BlScr	Blackout screen
e	External side	Top	Compartment above the thermal screen
in	Inside	Pipe	Heating floor Pipes
Out	Outside	GroPipe	Heating grow pipes
Sky	Sky	Cov	Cover

Table 2-1: Table summarising the meaning of subscripts in the differential equations

2-3 Generating the Data

2-3-1 Training Set

In order to have a meaningful comparison between models, they were all trained or fitted using the same data. The training set was chosen to be a mixture of artificially generated data, as well as regular operational data. The artificially generated data used several different excitation signals for the inputs and disturbances. This was done to sufficiently excite the nonlinear system. The three different excitation signals that were chosen were a pseudo random binary input sequence (PRBS), a random Gaussian signal (RGS) and an amplitude modulated pseudo random binary input sequence (APRBS). The first two signals are common excitation signals in system identification [36]. Amplitude-modulated PRBS is a common excitation signal specifically for nonlinear system identification [12]. The data that was generated with the different excitation signals was concatenated with the regular operational data. The total length of the training data was 27 840 time steps, which is roughly equal to three months of greenhouse simulation data.

The first 10 000 time-steps of the training data were driven by the three different excitation signals. From time-step 10 000 to time-step 27 840, the data was compromised of regular operations simulation data. The end of the training set was chosen to coincide with the end of a day, hence the choice of that number. The temperature of the air and the canopy of the training set can be seen in figure 2-3.

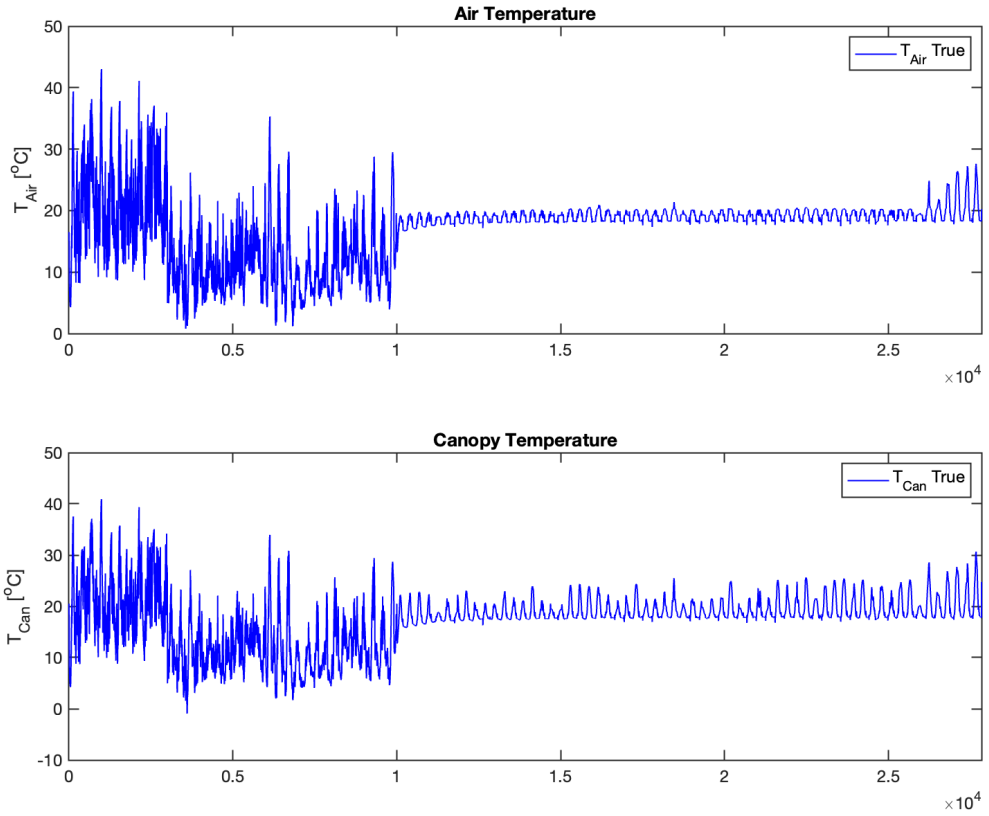

Figure 2-3: Air and canopy temperatures of the training set, combining artificial excitation and regular operations data

2-3-2 Test Set

For the purpose of evaluating the forecast accuracy, the test set was chosen to be a week long. The chosen week also contained a change in the dynamics from the first day to the rest of the days of the week. This choice was motivated by the idea that if a model could more accurately model the change of the dynamics, it would be an indication that the model has come closer to capturing the true dynamics of the sub-system. Furthermore, on day four of the test week, there is a stronger decoupling between the canopy and the air temperature, whereas on the other days, there are more closely correlated. If a models captures this decoupling, it is also an indicating that it captured the individual dynamics of the canopy temperature and the air temperature, rather than simply determining one based on the value of the other.

The canopy and air temperatures of the test week are shown in figure 2-4. For some comparisons, only the first day is used of the test set described in figure 2-4.

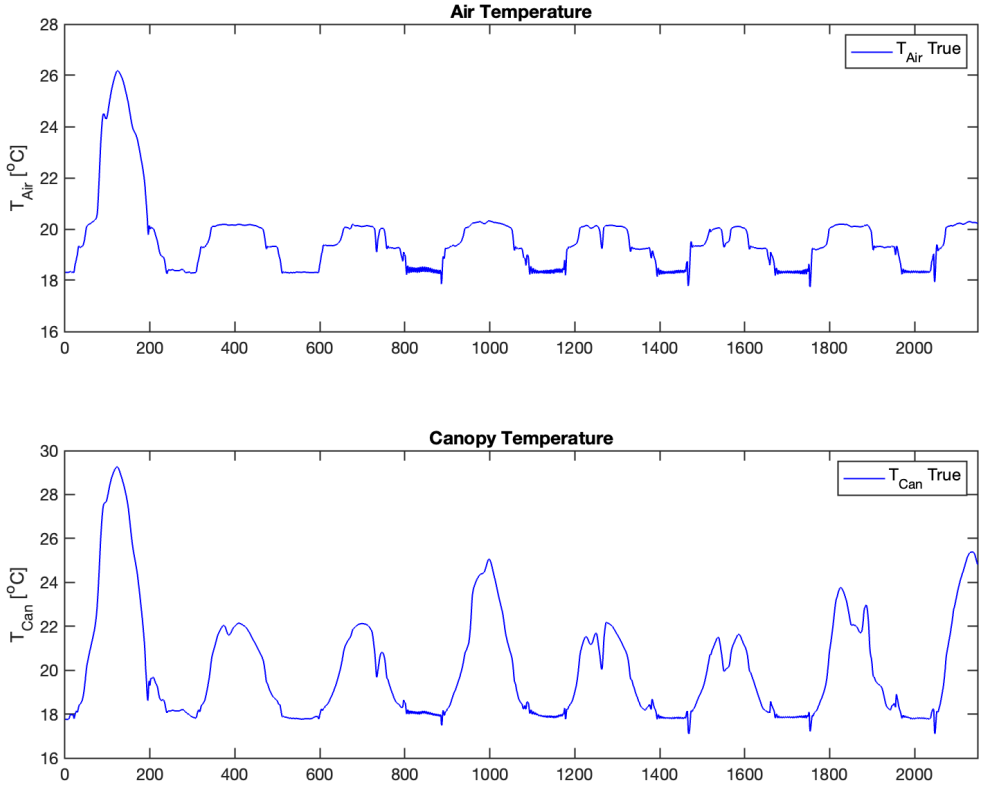

Figure 2-4: Air and canopy temperature of the chosen test week

All subsequent forecast models were trained on the same data, as well as tested on the same test set. The exception to the training set are the DeePC and eDeePC algorithms described in chapter 6. Further details on how those models were trained are detailed in that chapter. All data was scaled using a maximum absolute scaling before training and testing and un-scaled before the results were plotted.

Chapter 3

Linear Modelling

In this chapter, three common data-driven modelling approaches are presented and implemented using the greenhouse training data that was presented in the previous chapter. Their performance as forecast models on the test week will serve as a baseline for the non-linear data-driven models implemented in subsequent chapters. The three linear models that were implemented are an Auto-Regresssive with eXogenous input (ARX) model, a Dynamic Mode Decomposition with control (DMDc) model and a model based on subspace identification using the Multivariable Output-Error State-sPace (MOESP) algorithm. The ARX and the MOESP model were included because of they are established model structures in the field of system identification [36]. The DMDc model has also been widely applied but its inclusion is due to its close relation to the Koopman operator and the extended Dynamic Mode Decomposition. It will be used in subsequent chapters as a benchmark for the Koopman based approach.

3-1 ARX Model

3-1-1 Background

The first of the linear models that was tried is an Auto-Regressive with eXogeneous input model (ARX). It is a prediction error method that assumes an underlying model structure, given by equation 3-1.

$$y(k) = \frac{B(q)}{A(q)}u(k) + \frac{1}{A(q)}e(k) \tag{3-1}$$

Here, the term $e(k)$ is assumed to be a zero-mean white noise signal and the polynomials $A(q)$ and $B(q)$ are given by equations 3-2 and 3-3, with q being the time shift operator.

$$A(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n} \tag{3-2}$$

$$B(q) = b_1 q^{-1} + \cdots + b_{n_b} q^{-n} \tag{3-3}$$

The ARX predictor takes on the form as given by equation:

$$\widehat{y}(k) = B(q)u(k) + [1 - A(q)]y(k) \tag{3-4}$$

This predictor can be written as $\widehat{y}(k) = \phi(k)^{\mathrm{T}}\theta$ with θ and $\phi(k)$ given by equations 3-5 and 3-6, respectively.

$$\theta = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n_a} \mid b_1 & b_2 & \cdots & b_{n_b} \end{bmatrix}^{\mathrm{T}} \tag{3-5}$$

$$\phi(k) = \begin{bmatrix} y(k-1) & \cdots & y(k-n) \mid u(k-1) & \cdots & u(k-n) \end{bmatrix}^{\mathrm{T}} \tag{3-6}$$

The coefficients for the predictor are then found by minimizing the cost function given by 3-7.

$$J_N(\theta) = \frac{1}{N} \sum_{k=0}^{N-1} \Big(y(k) - \phi(k)^{\mathrm{T}}\theta \Big)^2 \tag{3-7}$$

3-1-2 Implementation

In order to fit this model for the chosen sub-system, the parameters n_a, n_b have to be determined first. n_a represents the number of previous outputs that are taken into consideration for the next step prediction, and can also be thought as the number of poles. The parameter n_b represents the number of previous inputs that are taken into account and is also referred to as the number of zeros. Furthermore, the parameter n_k is introduced and represents the delay after which the input affects the output. Essentially, this represents an additional shift to the input. Writing this as a difference equation without the shift operator q, this becomes:

$$\widehat{y}(k)) = b_1 u(k - n_k) + \ldots + b_{n_b} u(k - n_b - n_k + 1) - a_1 y(k-1) - \ldots - a_{na} y(k - n_a) \tag{3-8}$$

Since the system has two outputs, T_{Can} and T_{Air}, the ARX model can be written as a MIMO model by fitting a difference equation for each output. Each output is also taking the influence of the respective other output into account. The ARX model was fit using the system identification toolbox in MATLAB and the parameters n_a, n_b and n_k were determined by tuning the model. The best fit was achieved with values of $n_a = 2$, $n_b = 2$ and $n_k = 10$. This resulted in predictions over the test set as shown in figure 3-1.

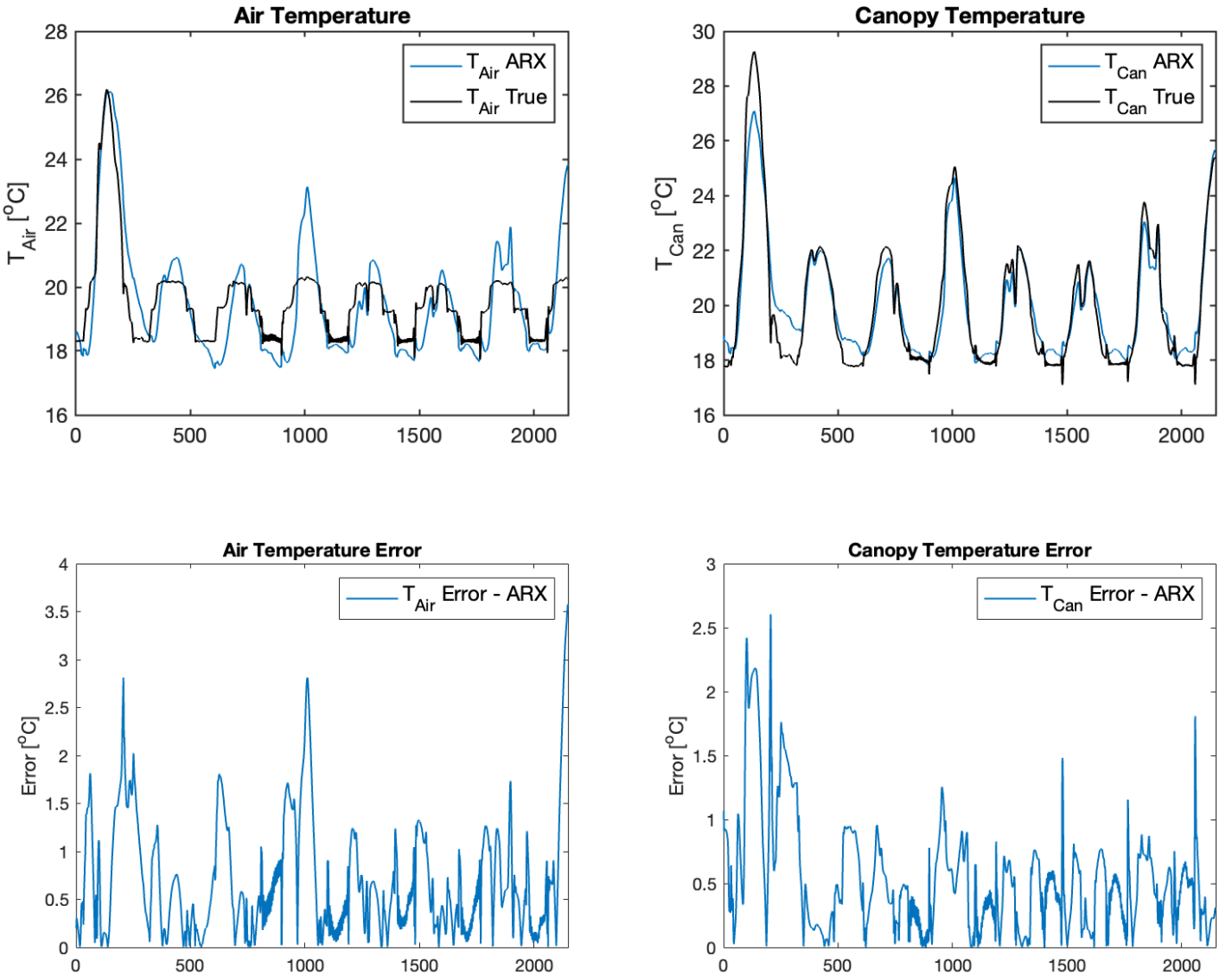

Figure 3-1: ARX predictor performance on the test set

3-2 Dynamic Mode Decomposition with control

3-2-1 Background

The Dynamic Mode Decomposition with Control (DMDc) is a method first described by [28]. It builds on the Dynamic Mode Decomposition initially developed by Schmid [32] as a tool to analyse and predict high dimensional complex systems. The DMDc attempts to find a discrete state space approximation, shown by equation 3-9 [28].

$$x(k+1) = Ax(k) + Bu(k) \tag{3-9}$$

The idea of DMD and DMDc is to collect snapshots of high dimensional input and output data, and estimate a reduced order model of the form that is shown in equation 3-9. It does this by utilizing the input matrix given by 3-10, the output measurements matrix 3-11 and the shifted output measurement matrix 3-12.

$$\Upsilon_{0,N-1} = \begin{bmatrix} | & | & & | \\ u(0) & u(1) & \dots & u(N-1) \\ | & | & & | \end{bmatrix} \tag{3-10}$$

$$X_{0,N-1} = \begin{bmatrix} | & | & & | \\ x(0) & x(1) & \dots & x(N-1) \\ | & | & & | \end{bmatrix} \tag{3-11}$$

$$X_{1,N} = \begin{bmatrix} x(1) & x(2) & \cdots & x(N) \end{bmatrix} \tag{3-12}$$

Note that the output matrices $X_{0,N-1}$ and $X_{1,N}$ contain snapshots of the state. This is because in the DMDc approach described by [28], the measured output is treated as equal to the state of the system. This means that the output matrix C is considered to be $C = I_n$ with n being the dimension of the reduced order system. This means there is the assumption that the state is directly observable.

Substituting the data matrices into the system representation given by 3-9 yields the data equation 3-13:

$$X_{1,N} = AX_{0,N-1} + B\Upsilon_{0,N-1} \tag{3-13}$$

In order to solve for the matrices A and B, the system can be rearranged to become:

$$X_{1,N} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} X_{0,N-1} \\ \Upsilon_{0,N-1} \end{bmatrix} = G\Omega \tag{3-14}$$

Where $\Omega = \begin{bmatrix} X_{0,N-1} \\ \Upsilon_{0,N-1} \end{bmatrix}$ matrices and $G = \begin{bmatrix} A & B \end{bmatrix}$. The analytical solution to find the best fit matrix G is given by 3-15.

$$G = \min_{G} \|X_{1,N} - G\Omega\|_F = X_{1,N}\Omega^{\dagger} \tag{3-15}$$

With high dimensional data matrices, dimensionality reduction in the form of a truncated singular value decomposition (SVD) is often applied to both Ω and $X_{1,N}$. The decompositions are denoted as $\Omega \approx \tilde{U}\tilde{\Sigma}\tilde{V}^*$ and $X_{1,N} \approx \hat{U}\hat{\Sigma}\hat{V}^*$, with truncation values p and r, respectively. The matrix $\tilde{U}$ can be divided into two matrices based on the dimension of input l and dimension of the state n into matrix $\tilde{U}_1 \in \mathbb{R}^{n\times p}$ and $\tilde{U}_2 \in \mathbb{R}^{l\times p}$, where $\tilde{U} = \begin{bmatrix} \tilde{U}_1^* & \tilde{U}_2^* \end{bmatrix}^T$. With this information, the reduced order approximations of A and B become:

$$\tilde{A} = \hat{U}^* X_{1,N} \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_1^* \hat{U} \tag{3-16}$$

$$\tilde{B} = \hat{U}^* X_{1,N} \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_2^* \tag{3-17}$$

3-2-2 Implementation

While originally designed for high dimensional systems, the DMDc algorithm was included in the comparison due to its close relation to the Koopman with input and control (KIC) approach, described in chapter 4. When DMDc was applied to the sub-system, the only two measurements were T_{Can} and T_{Air}. With an output dimension of $n = 2$ and input dimension $l = 8$, dimensionality reduction was not necessary to compute the analytical solution of equation 3-15.

Furthermore, Tikhonov regularization was applied. This is commonly done to reduce over-fitting and improve stability [26], [31]. The minimization problem described in 3-15, resulting in the formulation:

$$G = \min_{G} \|X_{1,N} - G\Omega\|_F + \lambda^2 \|G\|_F \tag{3-18}$$

Additionally, an alternative notation to the pseudo-inverse was used to avoid close to ill conditioned matrices in MATLAB. This notation uses the "/" operator also referred to as "mrdivide" function in MATLAB and is more computationally robust compared to the built-in pseudo-inverse function [13]. Using the notation as described in Garcia-Tenorio (2022) [13], the solution to the minimization problem in 3-15 becomes:

$$G = V/M \tag{3-19}$$

with auxiliary variables $V = X_{1,N}\Omega^T$ and $M = \Omega\Omega^T$.

The following algorithm summarizes the DMDc as it was implemented in MATLAB:

Figures 3-2 shows the predictions of the DMDc algorithm on the test set:

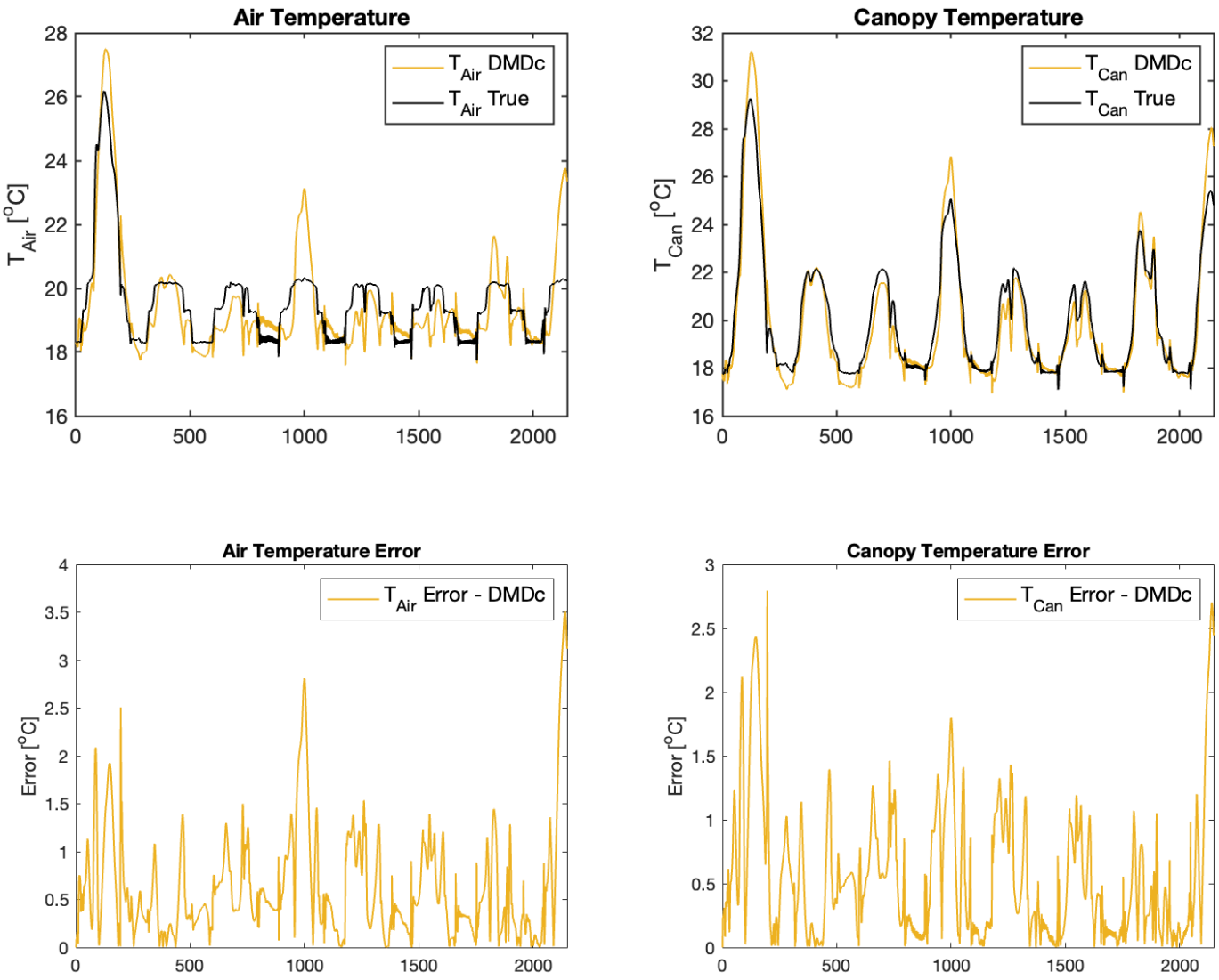

Figure 3-2: DMDc predictor performance on the test week

Algorithm 1: DMDc with Thikonov regularization and without dimensionality reduction

Input: $X_{1,N}$, $X_{0,N-1}$, $\Upsilon_{0,N-1}$
Output: $\tilde{A}$, $\tilde{B}$

1 Choose λ
2 Construct Ω by stacking $X_{0,N-1}$ and $\Upsilon_{0,N-1}$
3 Compute $V = X_{1,N}\Omega^T$
4 Compute $M = \Omega\Omega^T$
5 Compute the solution to the regularized least squares as $G = V/(M + \lambda I_n)$
6 Extract $\tilde{A}$ and $\tilde{B}$ from G through knowledge of n_x and n_u

3-3 Subspace Identification - MOESP

3-3-1 Background

The goal of subspace identification is to find a linear time-invariant system representation of the form given by

$$x(k+1) = Ax(k) + Bu(k), \tag{3-20}$$

$$y(k) = Cx(k) + Du(k), \tag{3-21}$$

This representation assumes a deterministic system. Subspace identification uses a finite number of samples of the input signal $u(k)$ and output signal $y(k)$ to find the system matrices (A, B, C, D) and initial state vector up to a similarity transform.

Several different algorithms for subspace identification exist, but the one that was chosen to be included in the comparison was the Multivariable Output-Error State-sPace (MOESP) algorithm as described in Verhaegen (2007) [36].

The MOESP algorithm finds the state space representation as shown in 3-20 and 3-21, by stacking the input and output data into block Hankel matrices of depth s:

$$Y_{0,s,N} = \begin{bmatrix} y(0) & y(1) & \cdots & y(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ y(s-1) & y(s) & \cdots & y(N+s-2) \end{bmatrix} \tag{3-22}$$

$$U_{0,s,N} = \begin{bmatrix} u(0) & u(1) & \cdots & u(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ u(s-1) & u(s) & \cdots & u(N+s-2) \end{bmatrix} \tag{3-23}$$

These hankel matrices can be substituted into the system representation given by 3-20 and 3-21 to formulate the Data Equation given by:

$$Y_{0,s,N} = \mathcal{O}_s X_{0,N} + \mathcal{T}_s U_{0,s,N} \tag{3-24}$$

where $X_{0,N} = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-1) \end{bmatrix}$. The so called extended observability $\mathcal{O}_s$ matrix is given by:

$$\mathcal{O}_s = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{s-1} \end{bmatrix} \tag{3-25}$$

And the matrix $\mathcal{T}_s$ is given by:

$$\mathcal{T}_s = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & & 0 \\ \vdots & & \ddots & \ddots & \\ CA^{s-2}B & CA^{s-3}B & \cdots & CB & D \end{bmatrix} \tag{3-26}$$

The strategy of MOESP is to first find the extended observability matrix up to a similarity transform T and to then solve for the remaining system matrices by solving a least squares problem. The extended observability matrix is found by stacking the output and input block Hankel matrices $Y_{0,s,N}$ and $U_{0,s,N}$ and applying an RQ-factorization:

$$\begin{bmatrix} U_{0,s,N} \\ Y_{0,s,N} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ R_{21} & R_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{3-27}$$

After this, an SVD is applied to the matrix R_{22} and the first n columns of the resulting matrix U_n will be equal to the extended observability matrix up to a similarity transform T.

$$R_{22} = \begin{bmatrix} U_n & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_n & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_n^{\mathrm{T}} \\ V_2^{\mathrm{T}} \end{bmatrix} \tag{3-28}$$

An extended proof for $\text{range}(U_n) = \text{range}(\mathcal{O}_s)$ can be found in Verhaegen (2007) [36]. The model order n is a parameter choice that is usually estimated based on the signular values of the SVD of R_{22}.

Once the extended observability matrix is identified up to a similarity transform T, it is trivial to determine the system matrices C and A up to similarity transformation T.

$$U_n = \mathcal{O}_s T = \begin{bmatrix} CT \\ CT\left(T^{-1}AT\right) \\ \vdots \\ CT\left(T^{-1}AT\right)^{s-1} \end{bmatrix} = \begin{bmatrix} C_T \\ C_T A_T \\ \vdots \\ C_T A_T^{s-1} \end{bmatrix} \tag{3-29}$$

Using the knowledge of the system matrices C_T and A_T, the system from equations 3-20 and 3-21 can be rewritten to result in:

$$\begin{aligned} y(k) =& C_T A_T^k x_T(0) + \left(\sum_{\tau=0}^{k-1} u(\tau)^{\mathrm{T}} \otimes C_T A_T^{k-\tau-1} \right) \mathrm{vec}\left(B_T\right) \\ &+ \left(u(k)^{\mathrm{T}} \otimes I_\ell \right) \mathrm{vec}\left(D_T\right). \end{aligned} \tag{3-30}$$

By defining a vector θ

$$\theta = \begin{bmatrix} x_T(0) \\ \mathrm{vec}\left(B_T\right) \\ \mathrm{vec}\left(D_T\right) \end{bmatrix} \tag{3-31}$$

And a matrix ϕ,

$$\phi(k)^{\mathrm{T}} = \begin{bmatrix} \widehat{C}_T \widehat{A}_T^k & \left(\sum_{\tau=0}^{k-1} u(\tau)^{\mathrm{T}} \otimes \widehat{C}_T \widehat{A}_T^{k-\tau-1} \right) & \left(u(k)^{\mathrm{T}} \otimes I_\ell \right) \end{bmatrix} \tag{3-32}$$

the remaining system matrices up to a similarity transform B_T, D_T and the initial condition $x_T(0)$ can be found by solving the minimization problem given by:

$$\min_{\theta} \frac{1}{N} \sum_{k=0}^{N-1} \left\| y(k) - \phi(k)^{\mathrm{T}} \theta \right\|_2^2 \tag{3-33}$$

3-3-2 Implementation

The MOESP algorithm was implemented using the subspace identification toolbox function *n4sid* in MATLAB. The model order was determined to be $n = 9$. The resulting performance on the test set can be seen in figure 3-3.

3-4 Comparing Linear Models

A comparison of the different models based on root mean square error (RMSE) can be seen in table 3-1.

	ARX	DMDc	MOESP
RMSE T_{Air}	0.9083	0.8386	0.5873
RMSE T_{Can}	0.7004	0.7569	0.6034

Table 3-1: Comparison of RMSE values of linear models on the test week

The performance of all linear models was compared on the same test set. The ARX model and the DMDc model had similar performance and the model with the lowest prediction errors on the test set was the MOESP predictor. The main difference between the DMDc model and the ARX model is the inclusion of delay embeddings, as well as the inclusion of regulariztation for the DMDc approach. The state of the DMDc model was equal to the dimension of the output and did not include any delay embeddings. Furthermore, a time-shift n_k for when the input affects the output was included in the ARX model. However, little significant different exists between the two approaches, when in comes to the prediction error on the test set.

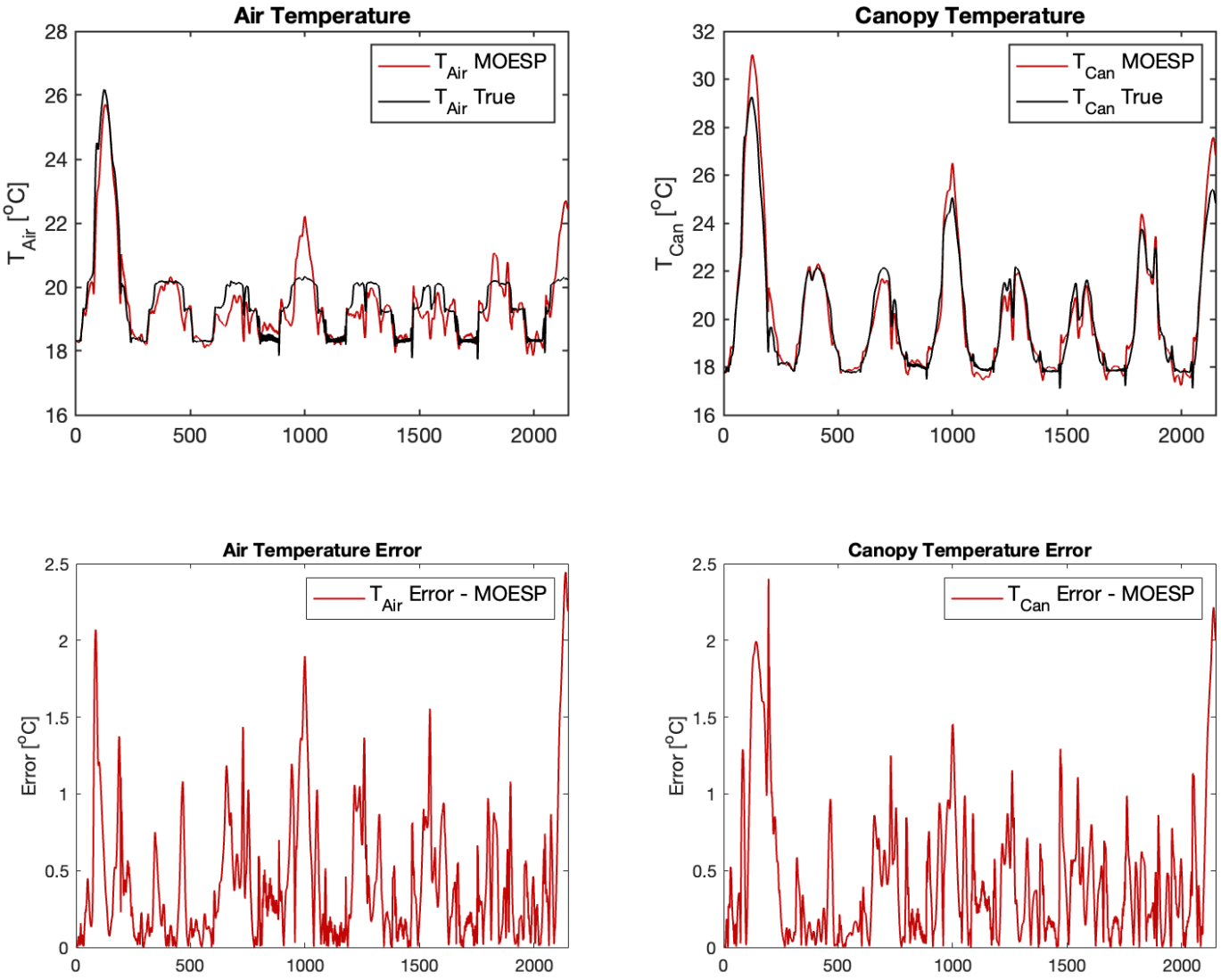

Figure 3-3: MOESP predictor performance on the test set

The best predictions were found using the MOESP model, which models a higher-dimensional linear system (in this case $n = 9$). In contrast, the DMDc approach only attempts to estimate a two-dimensional system based on the outputs. The higher dimensionality of the MOESP approach results in better predictions. The underlying system is non-linear and only a sub-system is being modelled, with the full system that determines the dynamics having a higher dimensionality. Thus, a higher dimensional linear state space representation will come closer to the true, high dimensional nonlinear model, than a lower dimensional state space representation as in the case of the DMDc model.

Since the ARX model is also only using the two dimensional outputs, with an additional delay embedding to propagate the output, the performance is similar to the two dimensional representation of the DMDc approach.

A complete comparison of the different linear modelling approaches on the test set can be seen in figure 3-4. The figure illustrates that the subspace identification using the MOESP algorithm performed the best. However, the MOESP model still results errors on the test set of over $1.5C^{\circ}$. This provides motivation to implement models that have non-linearities encoded in the model structure to improve prediction errors.

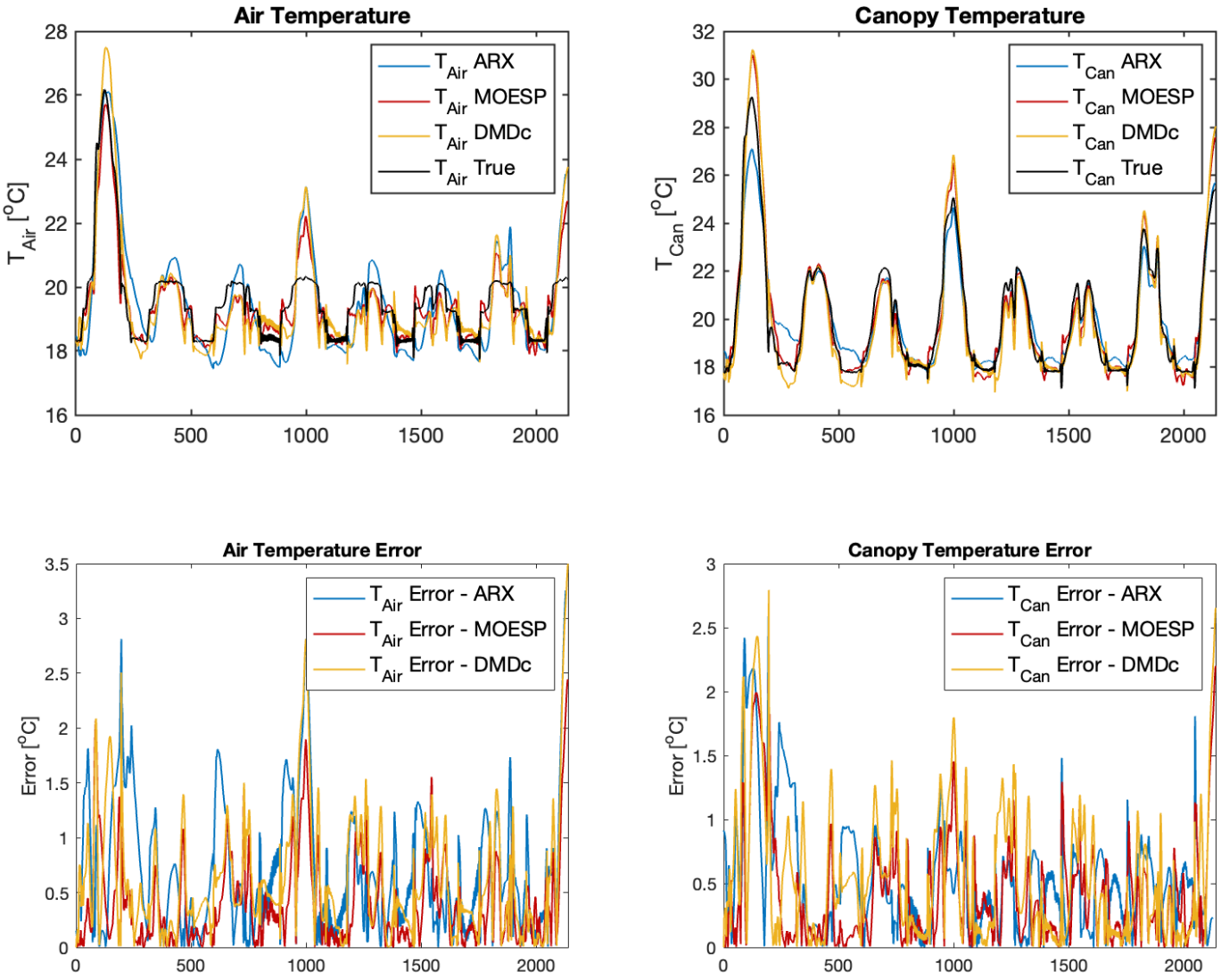

Figure 3-4: Comparing all linear predictors performance on the same test set

Chapter 4

Long Short-Term Memory Models

Long Short-Term Memory models are a type of gated recurrent neueral network architecture that is commonly used for time series prediction. It was originally introduced by Hochreiter (1997) [15]. Like other recurrent neural networks it is a type of black-box model that contains non-linearities in the form of non-linear activation functions that are combined with linear weights. Because of their proven capability to predict time series data well, and because of its successful application in greenhouse climate prediction [34], it was chosen to be included in the comparison.

4-1 LSTM Cell Architecture

Several variations of LSTMs exist. The one that was used in this thesis is the implementation that the keras python library is using. This implementation follows the original variation presented in [15]. Figure 4-1 shows the internal structure of an LSTM cell with its different gates.

The innovation of LSTMs is the introduction of a memory cell variable c_t and gating control signals i_t, f_t and o_t. The propagation of memory overcomes the vanishing gradient problem during training and captures long term dependencies [30]. The memory cell of an LSTM functions in the following manner:

A simple recurrent neural network (RNN) takes the input vector s_t, the (previous) hidden state h_t at time t and computes an intermediate memory cell variable $\tilde{c}_t$. The weighted sum of this intermediate cell state and the previous cell state c_{t-1} is then added to the weighted sum of the previous state c_{t-1} and the forget gating signal f_t to produce the new cell state c_t. The new hidden state is then a weighted sum of the output signal o_t and the hyperbolic tangent of the new cell state $g(c_t)$. Equations 4-1, 4-2 and 4-3 show the calculation of the intermediate memory state $\tilde{c}_t$, the memory state c_t and the hidden state h_t, respectively.

$$\tilde{c}_t = g\left(U_c h_{t-1} + W_c x_t + b_c\right) \tag{4-1}$$

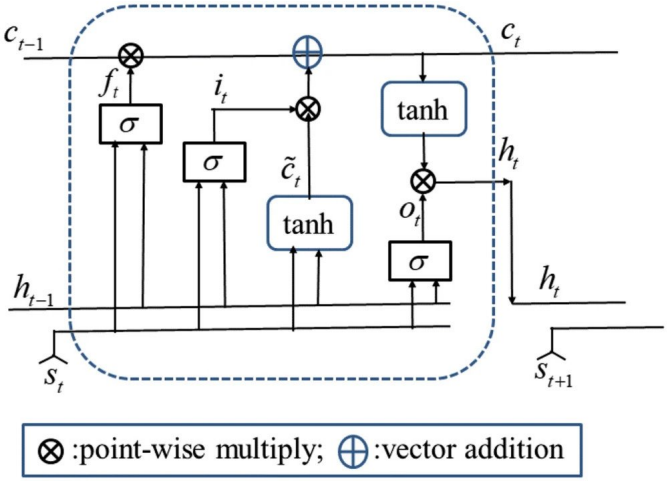

Figure 4-1: Schematic of an LSTM cell [30]

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{4-2}$$

$$h_t = o_t \odot g\left(c_t\right) \tag{4-3}$$

The weighted sum from equations 4-2 and 4-3 is an element-wise (Hadamard) multiplication and is denoted by $\odot$.

The three control gating signals i_t, f_t and o_t correspond to the input, forget and output gate of the LSTM cell. Their equations are described by 4-4, 4-5 and 4-6.

$$i_t = \sigma\left(U_i h_{t-1} + W_i x_t + b_i\right) \tag{4-4}$$

$$f_t = \sigma\left(U_f h_{t-1} + W_f x_t + b_f\right) \tag{4-5}$$

$$o_t = \sigma\left(U_o h_{t-1} + W_o x_t + b_o\right) \tag{4-6}$$

Like other recurrent neural networks, the weights of an LSTM are trained by using back-propagation through time (BPTT). For the three gates and the memory cell, the werights to be trained are $W_i, U_i, b_i, W_f, U_f, b_f, W_o, U_o, b_o, W_c, U_c$, and b_c. These are the parameters that are updated at each training iteration.

4-2 Default Single Layer LSTM

The application to the green house system followed the example as has been described by [17], [23], [34]. The implementation outlined in the work by Liu et al. (2022) [23] served as the basis for the implementation in this thesis. Their approach to forecasting future climate

trends is rooted in the notion that short-term climate variations play the most relevant role in the overall climate trajectory. From this notion, a sliding window approach was implemented.

In the implementation of Liu et al (2022) [23] a window size of five was chosen. This meant that the input signals from time-step one to five were used to predict the climate at time-step six. After that the window would slide and the inputs from time-step two to six were used to predict the climate at time-step seven.

The length of this window size is a variable that needs to be redefined for each use case. In this work, the implementation had a sliding window size of 36, corresponding to three hours of previous input data. Figure 4-2 depicts the sliding window approach as it was adapted to for this thesis. Note that the input vector s_t from figure 4-1 has now been changed to u_k with k being the time-step. This is to be in accordance with the rest of the notation used earlier and to illustrate that this is the same input vector that was used for the linear methods. Similarly, the hidden state h_t was directly used as the output without any further embedding layers and thus is referred to as y_k.

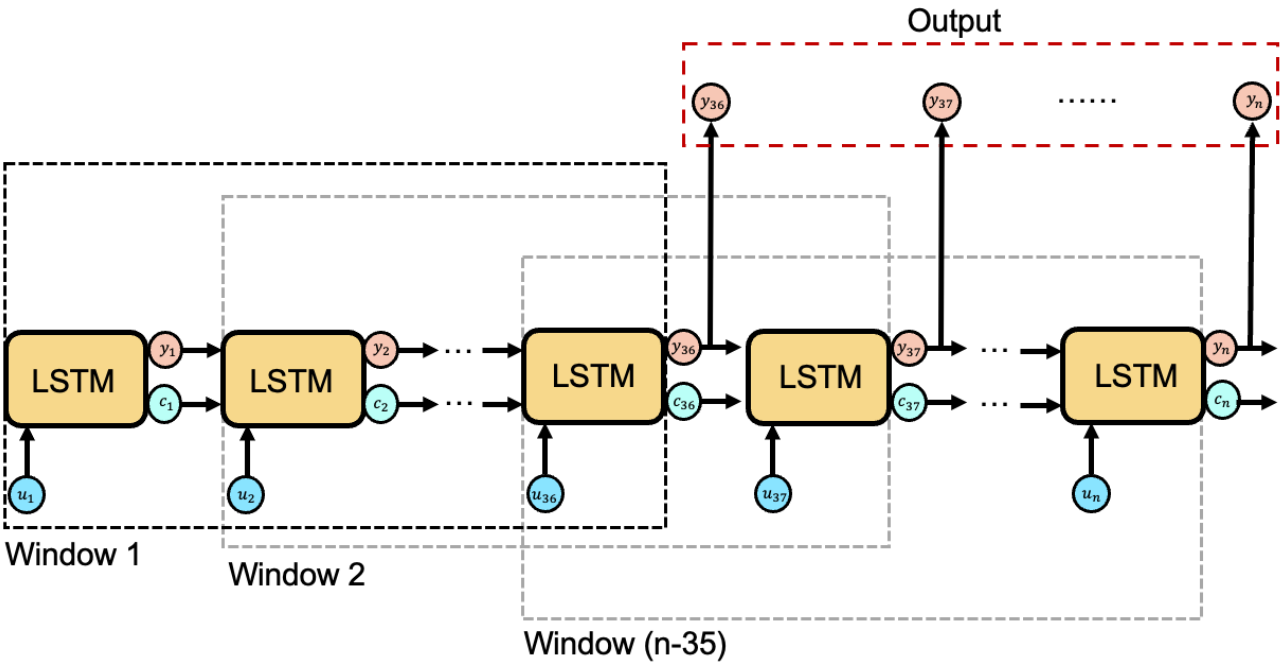

Figure 4-2: Single layer sliding window LSTM approach based on Liu et al (2022) [23]

The weights of the architecture described by figure 4-2 were trained on the data described in 2-3-1 for ten epochs using BPTT. The resulting predicitons on the test set are shown in figure 4-3. While the error on the first day still is above two degrees, similar to the linear methods, the LSTM performs better as the week continues and stays below one degree of error for the canopy temperature and within 1.5 degrees error for the air temperature. Similarly to the previous models, the air temperature predictions are worse than the canopy temperature predictions.

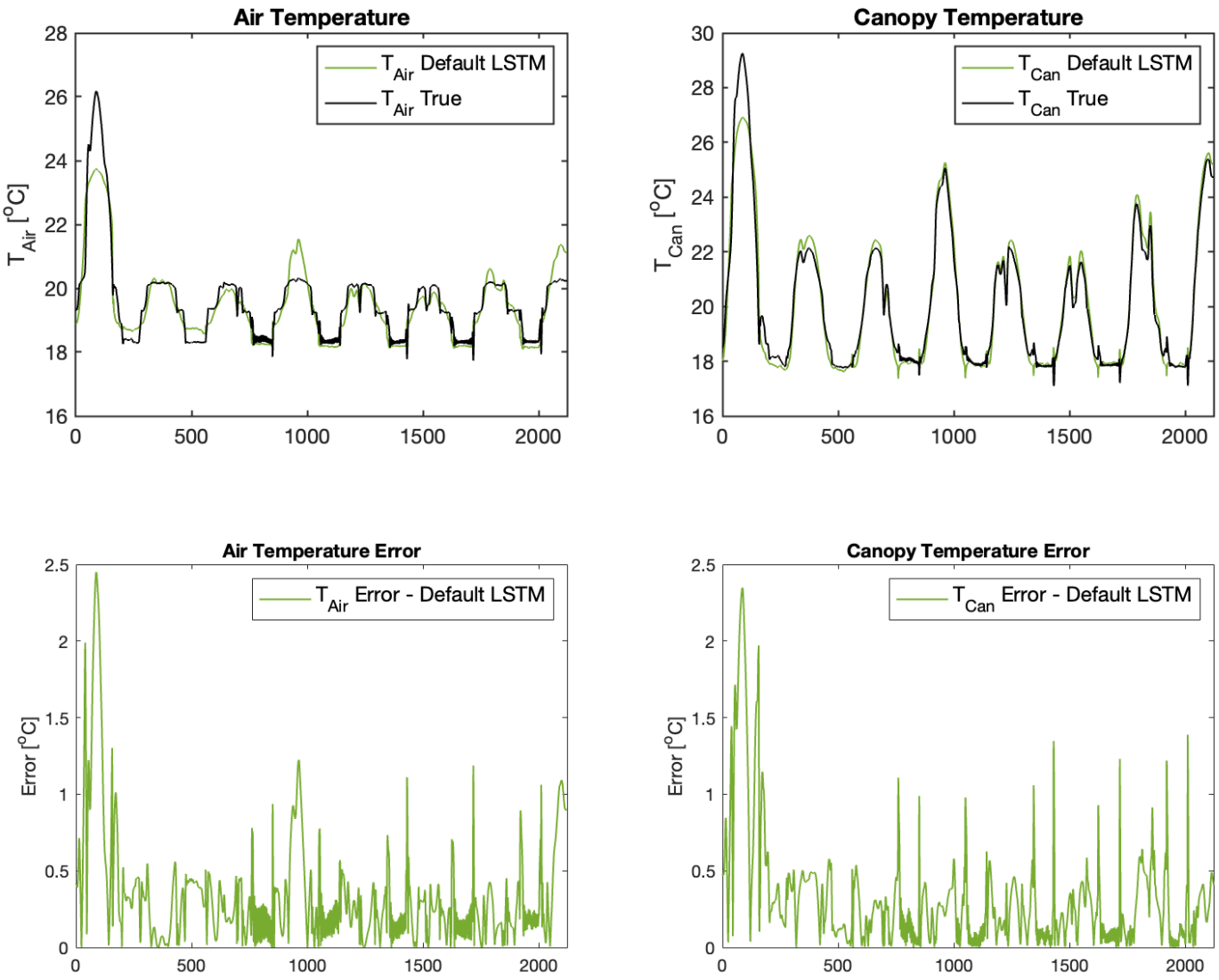

Figure 4-3: Single layer LSTM predictions on the test set

4-3 Stacked LSTM

The architecture presented in figure 4-2 depicts the default, single layer LSTM architecture. However, in order to further improve prediction results, different architectures were implemented. Stacking two LSTMs has proven to be able to capture more complex temporal dependencies [39]. In order to better capture the complex dynamics of the greenhouse system, a stacked architecture was implemented using the same sliding window approach that was applied in the single layer architecture. The same window size of 36 was chosen. An illustration of how the stacked architecture works, can be seen in figure 4-4. The hidden state of the first layer is used as the input vector for the second layer. Staying in the same notation for inputs and outputs, the hidden state of the first layer at time k is denoted by y_k^0 and the memory state at time k denoted by c_k^0. Consequently, the hidden state of the second layer at time k is referred to as y_k^1 and the memory state of the second layer at time k as c_k^1. The input vector at time step k is denoted the same way as in the single layer LSTM as u_k.

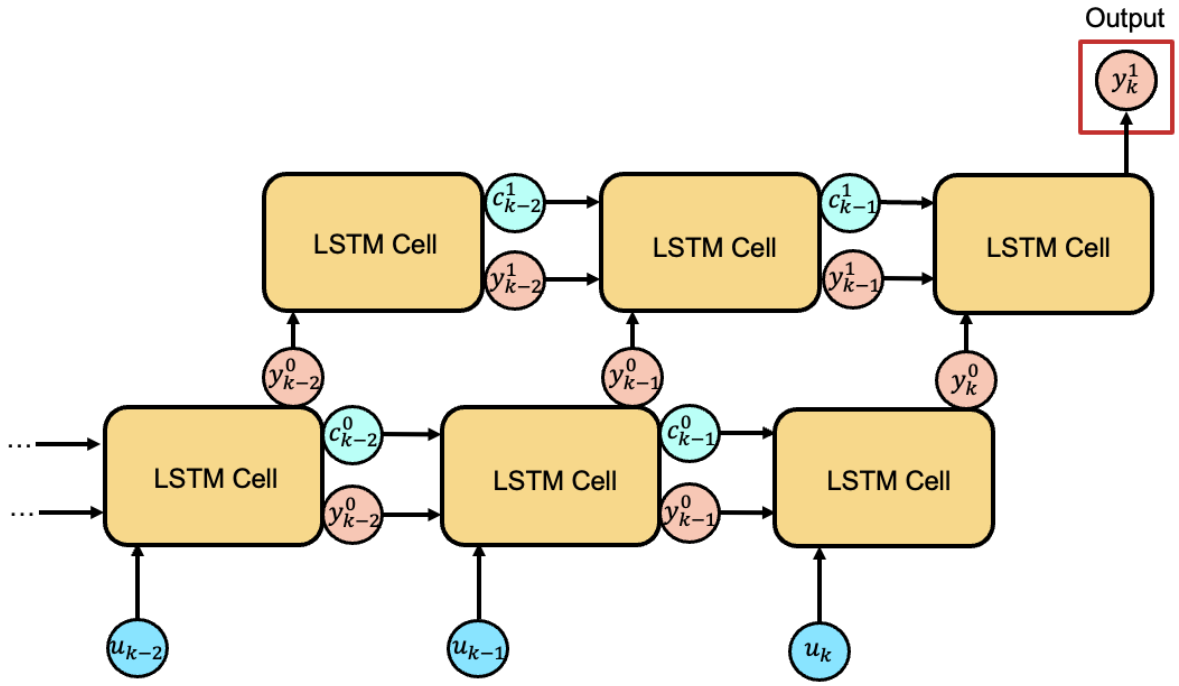

Figure 4-4: Stacked LSTM Architecture

Because this stacked LSTM architecture now contained more parameters that require training, the model was now trained for 100 epochs. This stacked architecture was better able to capture the changing dynamics from the first to the second day and overall had lower prediction errors over the entire week as shown in figure 4-5. The prediction error stayed within half a degree for all but the first days of the test set. It also performed significantly better on the first day when compared to the single layer LSTM.

The stacked LSTM architecture also outperforms the best linear predictor as a forecast model for the week ahead prediction on the test set. A summary of the RMSE values over the test sets between the MOESP model, default LSTM and stacked LSTM model can be seen in table 4-1.

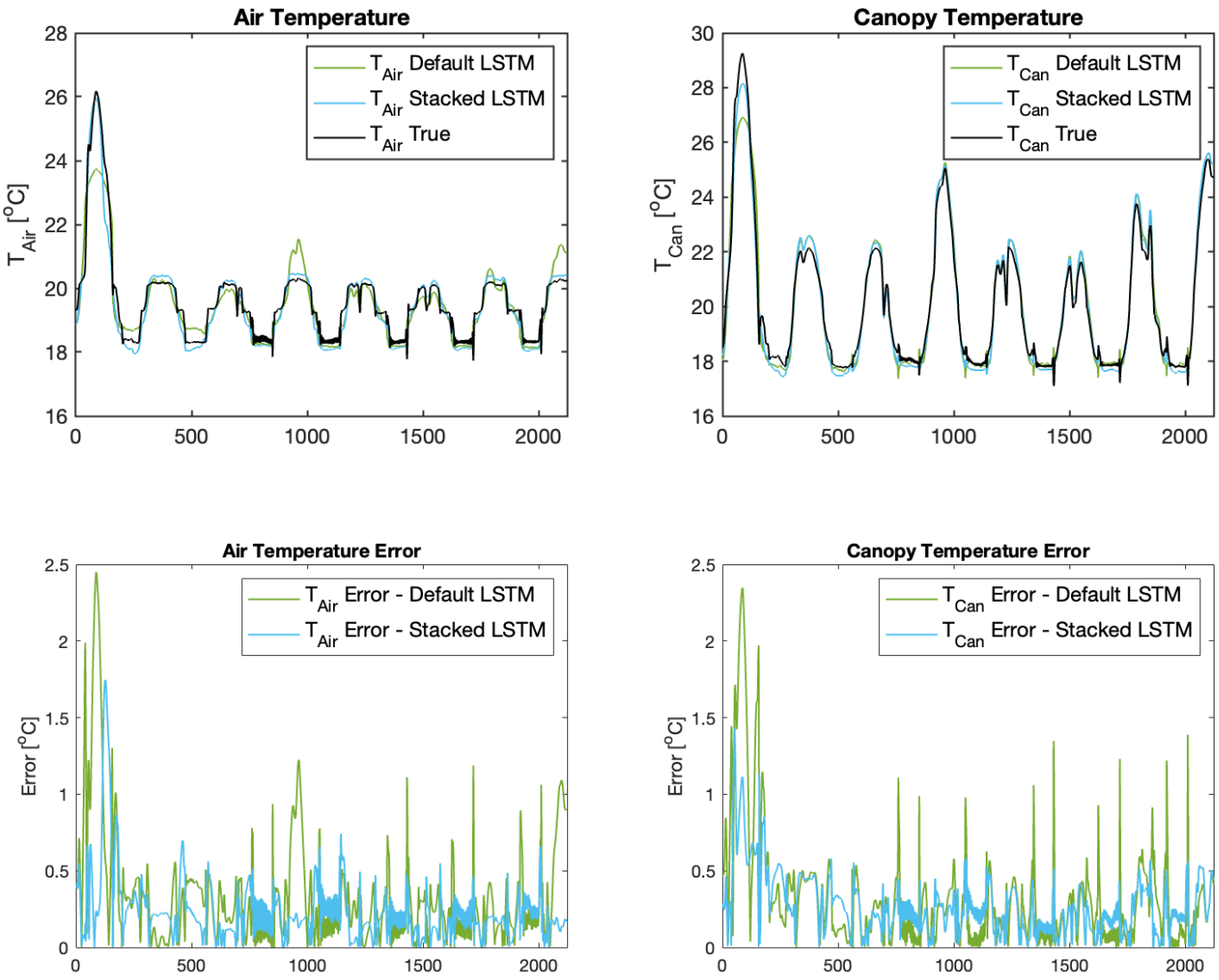

Figure 4-5: Stacked and Single LSTM Architecture Results on Test Trajectory

	MOESP	default LSTM	stacked LSTM
RMSE T_{Air}	0.5873	0.5055	0.3234
RMSE T_{Can}	0.6034	0.4916	0.3363

Table 4-1: Comparison of RMSE values of LSTM architectures with MOESP model on the test set

Table 4-1 and figure 4-5 show that a stacked LSTM architecture can increase the accuracy of greenhouse climate forecasts over single layer architectures that have been implemented in the literature [17], [23], [34]. This is because a stacked architecture can capture more complex dynamics by essentially increasing the memory of the system [39]. Table 4-1 also illustrates the advantage of LSTM models over the implemented linear models.

4-4 Multiple parallel LSTMs

In an attempt to come up with a further improved LSTM architecture, another approach was devised. The intuition behind this approach was that the dynamics are different throughout the day and go through different phases. The dynamics during the day are different when compared to the dynamics of the night and in the morning and in the evening the rising and falling temperature dynamics are different from the middle of the day and the middle of the night. Hence, the idea was to train separate LSTMs that each capture the behaviour of different sections of the day cycle.

The full day in the simulation consists of 288 time-steps and four main phases of the day were identified. Each phase was chosen to be of equal length and determined to be 72 time-steps long. These can be seen in figure 4-6. The first phase is the rising of the temperature , the second is the evolution during the middle of the day. In the third phase, the temperature decreases again and in the last phase, the temperature remains at a low level during the night time.

The training data was then split up according to these four phases and a separate LSTM was trained for each phase, resulting in four different LSTMs. For each LSTM, the structure that was used was the default single layer architecture, outlined in section 4-2. This also meant that the same sliding window approach as was outlined in figure 4-2 was applied. The same window length of 36 was chosen. An illustration for this multiple LSTM architecture can be seen in figure 4-7. In the figure, $u(k, 36)$ denotes the 36 previous inputs that correlate to output $y(k)$ at time k. This means output $y(k)$ is predicted using the $k - 36$ previous inputs up to and including input at time k.

Note that each LSTM was trained on data that only consisted on the respective phase of the day and the predictions are also only limited to the part of the day that the LSTM was trained for. The first LSTM only predicts from the first to the 72nd time-step of the day, the second from time-step 73 to 144 and so on. By training each LSTM only on a specific part of the day, the goal was that this would result in more specialised models that would be better at predicting the dynamics of each phase of the day. This also meant that each LSTM would have less training data, since now the same training data was split up between four models.

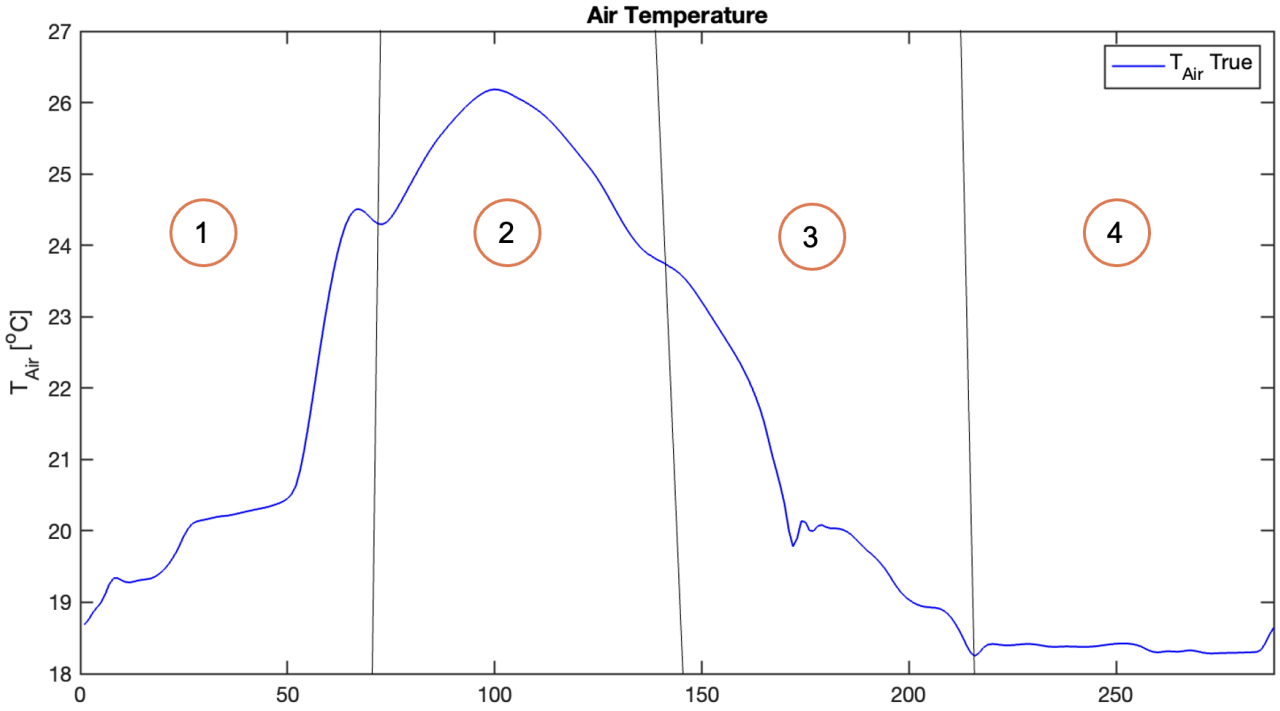

Figure 4-6: Division of the day into similar dynamics

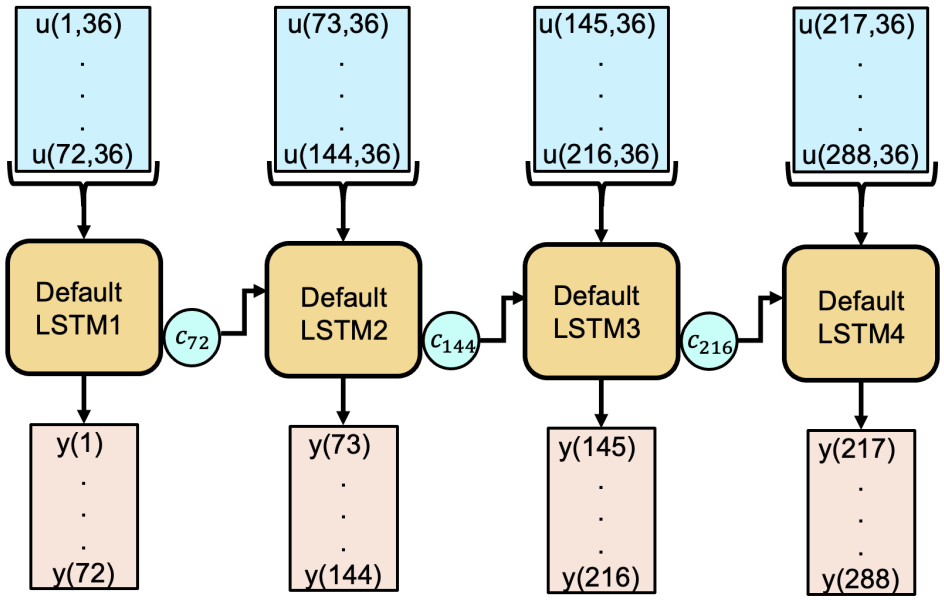

Figure 4-7: Multiple parallel LSTM architecture based on similar dynamics of the day cycle

After the temperatures for each phase of the day are predicted, the predictions are concatenated together to form one trajectory. Due to the fact that the predictions of each phase of the day are done separately, the predictions can contain jumps when concatenating predictions together. In order to avoid this, the internal state from one LSTM was passed onto the internal state of the LSTM for the following phase. This is possible because all LSTMs have the same internal structure. It is illustrated in figure 4-7 with the arrows that pass on the internal state c_i from one LSTM to the next.

The resulting predictions of this alternative architecture can be seen in figure 4-8. They are compared to the predictions of the default single layer LSTM presented in section 4-2. As can be seen in this figure, the multiple LSTM implementation still contains discontinuous jumps even though the internal cell states have been passed on. These discontinuous jumps are due to the fact the internal gates of the LSTM described by equations 4-4, 4-5 and 4-6 also contain the non transferable weights and biases denoted by $W_i, U_i, b_i, W_f, U_f, b_f, W_o, U_o$ and b_o. These are determined during training and due to the fact that each LSTM is trained on separate training data, the weights are different for each LSTM. While transferring the state can bring the predictions closer together and reduce the jumps, it cannot eliminate the discontinuity. Furthermore, figure 4-8 shows little to no improvement over the default implementation. Using the same amount of data, the predictions do not appear to be further improved by dividing the data up and training specialised LSTMs for each phase of the day.

While this alternative approach using multiple parallel LSTMs was proposed to further improve the prediction accuracy, no significant improvement was achieved and the introduced discontinuity issue is not resolvable. The most accurate LSTM predictor remains a stacked LSTM architecture.

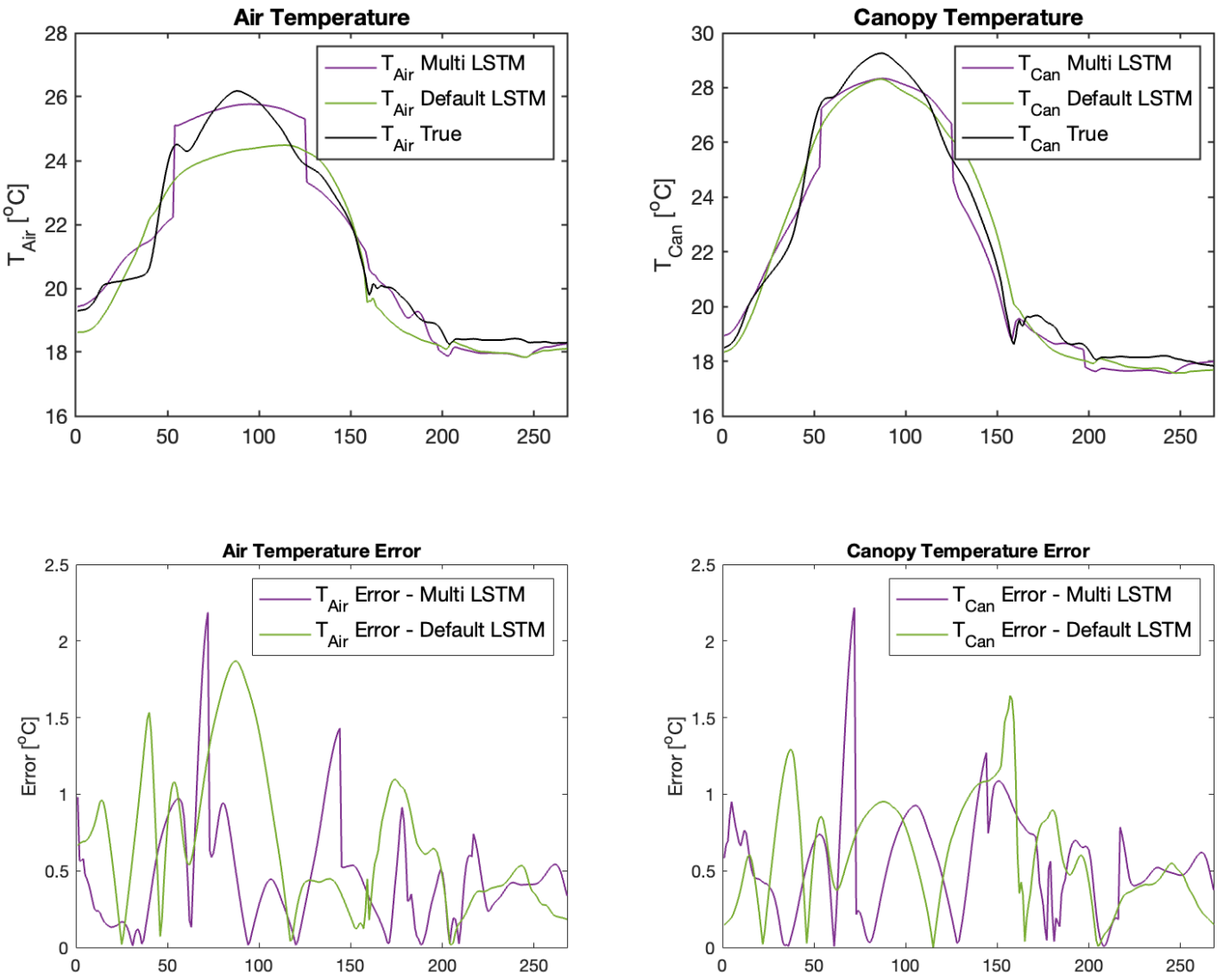

Figure 4-8: Multiple parallel LSTM vs Default LSTM on the same day

Chapter 5

Modelling with Koopman Theory

Koopman operator theory has gained attention in recent years as a novel method to model nonlinear dynamics. Through data-driven algorithms such as the extended dynamic mode decomposition (eDMD) [37], it has found applications in the aerospace sector for the control of cubesats [6], controlling soft robotics [3], quad-rotors [41] and many more. Koopman based approaches focus on a crucial idea: they elevate the original state space variables using a nonlinear transformation. This process shifts them into a lifted space where the evolution of the transformed state space becomes linear. A globally linear representation of the system can then be used for control or forecast models and promises to yield better prediction results for nonlinear systems than simple linear models. Finding the transformation that will yield a linear evolution in the lifted space remains the biggest challenge when applying Koopman theory in practice [18].

5-1 Koopman Theory

The Koopman operator framework provides a different perspective on dynamical systems. The following will focus on the discrete-time dynamical system descriptions since this is also the case of application. The description and equations for this background are taken from Mauroy et al. [24].

Assuming a discrete time system that evolves through a nonlinear transformation $S : X \to X$, where the state space X is a finite-dimensional euclidean space, one can describe the system through its trajectories $\left\{S^k(x)\right\}_{k=0}^{\infty}$, with x being an initial condition. The innovation of the Koopman operator framework is that a system can also be viewed though its output functions $g : X \to \mathbb{C}$, also referred to as observable functions and on the trajectories $\left\{f\left(S^k(x)\right)\right\}_{k=0}^{\infty}$. The evolution of all observable functions is defined by the (linear) Koopman operator.

definition 5-1.1. *Consider a hilbert space of observables $g : X \to \mathbb{C}$. The Koopoman operator $\mathcal{K}_S : \mathscr{H} \to \mathscr{H}$ associated with the map $S : X \to X$ is defined through the composition*

$$g(x_{k+1}) = \mathcal{K}_S g(x_k) = g \circ F(x_k) \tag{5-1}$$

The advantage that the Koopman operator framework provides it that it results in a globally linear system description of a nonlinear systems [37]. The drawback is that, the Koopman operator is inifinte-dimensional. To overcome this challenge and to be useful in practice, the goal often is to find a finite dimensional subset of observable functions that can either fully describe the system dynamics or converges with increasing order such that a truncation at a sufficiently high dimension of the lifted state will yield a good approximation of the system dynamics. Several methods for how to find these observable functions have been proposed [18], [4]. This remains an open research challenge when applying Koopman operator theory [18].

5-1-1 Extended Dynamic Mode Decomposition

A commonly applied method for approximating the Koooopman operator has been the extended dynamic mode decomposition (eDMD), proposed initially by Williams et al. [37] and further extended for control (eDMDc) by Korda et al.[21]. As the name suggests, eDMD is closely related to the dynamic mode decomposition and the dynamic mode decomposition with control, described in the section 3-2. However, instead of directly using measurements of the state to approximate a linear mapping to the next time-step, it takes the state measurements and applies a transformation using the (nonlinear) observables first. This results in a new, augmented state vector:

$$\mathbf{g}\left(x_k\right) = \begin{bmatrix} g_1\left(x_k\right) \\ g_2\left(x_k\right) \\ \vdots \\ g_n\left(x_k\right) \end{bmatrix} \tag{5-2}$$

Here, x_k is the measurement at time k and represents the state space vector. Because eDMD is related to DMD and DMDc, this approach also assumes direct measurements of the state are available.

This augmentation to the state measurement, also referred to as lifting, is applied to the time-series of measurements:

$$\mathbf{g}(X_{0,N-1}) = \begin{bmatrix} | & | & & | \\ \mathbf{g}\left(x_0\right) & \mathbf{g}\left(x_1\right) & \cdots & \mathbf{g}\left(x_{N-1}\right) \\ | & | & & | \end{bmatrix} \tag{5-3}$$

From here on, the algorithm is the same as the DMD. Using a time shift, a second lifted data matrix composed of shifted snapshots is constructed:

$$\mathbf{g}\left(X_{1,N}\right) = \begin{bmatrix} | & | & & | \\ \mathbf{g}\left(x_1\right) & \mathbf{g}\left(x_2\right) & \cdots & \mathbf{g}\left(x_N\right) \\ | & | & & | \end{bmatrix} \tag{5-4}$$

The goal is to find a best fit approximation of the Koopman operator $\mathcal{K}$, that describes the evolution in the lifted space. The hope is that if the observables were chosen correctly, a best fit linear mapping between these two lifted time-series should be approximating the Koopman

operator. This approximation is denoted by K instead of $\mathcal{K}$. The best fit linear approximation that evolves $\mathbf{g}(X)$ to $\mathbf{g}(X')$ is found by solving the least squares problem:

$$\min_{K} \|\mathbf{g}(X_{1,N}) - K\mathbf{g}(X_{0,N-1})\|_F \tag{5-5}$$

Here, $\|\cdot\|_F$ is the Frobenius norm. The analytical solution to this problem is given by:

$$K = \mathbf{g}(X_{1,N})\,\mathbf{g}(X_{0,N-1})^{\dagger} \tag{5-6}$$

With † being the Moore-Penrose pseudoinverse.

In practice, this linear regression often includes a regularization term in order to prevent overfitting of the approximation of Koopman operator K [9].

5-1-2 Generalizing Koopman Theory for Control - KIC

Several expansions for adapting eDMD for control. The most closely related to DMDc is eDMDc, as described in [21]. This has been the framework commonly applied to estimate the Koopman operator to systems with inputs [6], [3], [41]. The eDMDc algorithm treats inputs linearly with respect to the system dynamics. Building on this work, Proctor et al. [29] developed a generalized framework to also include inputs in the nonlinear observables. This framework is called Koopman with inputs and control and is referred to henceforth as KIC. As described in section 2-2 of chapter 2, the inputs and disturbances are treated as exogenous input signals that evolve independently. Using this information and the KIC framework described in [29], the definition of the Koopman operator is as follows:

$$\mathcal{K}\mathbf{g}(x_k, u_k) \triangleq \mathbf{g}(f(x_k, u_k), 0) \tag{5-7}$$

This formulation explicitly allows for mixed observables, as for example $g(x, u) = xu$. This is relevant to the application in the greenhouse since a lot of the dynamics within the greenhouse climate are nonlinearly evolving as a function of external disturbances, such as the outside temperature T_{Out} or input signals, such as the temperature of the heating pipe T_{Pipe}.

In order to simplify notation, the observable functions dependent on state measurement x at time-step k will be denoted as $\mathbf{z}_{x,k} = \mathbf{g}_x(x_k)$. Mixed observables, dependent on x and u at time-step k, the notation becomes $\mathbf{z}_{xu,k} = \mathbf{g}_{xu}(x_k, u_k)$ and observables only dependent on the input u at time-step k are now denoted by $\mathbf{z}_{u,k} = \mathbf{g}_u(u_k)$. Using the new notation, equation 5-7 can be rewritten to result in:

$$\begin{bmatrix} \mathbf{z}_{x,k+1} \\ \mathbf{z}_{xu,k+1} \\ \mathbf{z}_{u,k+1} \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} \mathbf{z}_{x,k} \\ \mathbf{z}_{xu,k} \\ \mathbf{z}_{u,k} \end{bmatrix} \tag{5-8}$$

Since the goal is not to predict the evolution of the input or the mixed observables, the formulation of equation 5-8 can be simplified:

$$\begin{bmatrix} \mathbf{z}_{x,k+1} \\ \mathbf{z}_{xu,k+1} \\ \mathbf{z}_{u,k+1} \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_{x,k} \\ \mathbf{z}_{xu,k} \\ \mathbf{z}_{u,k} \end{bmatrix} \tag{5-9}$$

The approximation of the Koopman operator from this KIC framework thus results in the following formulation:

$$\mathbf{z}_{x,k+1} = K\mathbf{z}_k = K \begin{bmatrix} \mathbf{z}_{x,k} \\ \mathbf{z}_{xu,k} \\ \mathbf{z}_{u,k} \end{bmatrix} \tag{5-10}$$

The approximation of the Koopman operator in the KIC framework can be found in a similar way as with the eDMD algorithm, as presented in the analytical solution in equation 5-6. This means that a linear regression including regularization can be used in the KIC approach as is done in the eDMD algorithm.

5-2 Implementation

5-2-1 Choice of Lifting Functions

The choice of the right lifting functions remains an open challenge [22]. Trying to find a good basis of lifting functions by using common choices such as radial basis functions or different polynomial series did not yield any acceptable results. Thus, the choice was made to choose lifting functions based on the underlying equations of the differential equations that dominate the dynamics of the temperature.

The underlying differential equations for the two temperatures are given by equation 2-10 and 2-11. The non-zero components of the differential equation for T_{Air} are functions of states of the system. Stacking all heat transfers of T_{Air}, which are dependent on other states and inputs, yields the following expression:

$$\begin{bmatrix} H_{\text{CanAir}}(T_{Can}, T_{Air}) \\ H_{\text{PipeAir}}(T_{Pipe}, T_{Air}) \\ R_{GlobSunAir}(u_{ThScr}, d_{iGlob}) \\ H_{\text{AirFlr}}(T_{Flr}, T_{Air}) \\ H_{\text{AirThScr}}(u_{ThScr}, T_{ThScr}, T_{Air}) \\ H_{\text{AirOut}}(T_{Air}, T_{Out}, d_{wind}) \\ H_{\text{AirTop}}(T_{Air}, T_{Top}, u_{ThScr}) \\ H_{\text{GroPipeAir}}(T_{Air}, T_{GroPipe}) \end{bmatrix} \tag{5-11}$$

Some heat exchanges depend on states that are not included in the sub-system description that was outlined in section 2-2. These terms are $H_{\text{AirTop}}(T_{Air}, T_{Top}, u_{ThScr})$, $H_{\text{AirFlr}}(T_{Flr}, T_{Air})$ and $H_{\text{AirThScr}}(u_{ThScr}, T_{ThScr}, T_{Air})$. These functions were excluded from the set of observable functions, since they would have required information that was not available to the sub-system.

As was mentioned before, in this approach, the output is assumed to be equal to the state. By sticking with the same notation, this means that $x_k = y_k = \begin{bmatrix} T_{Can} \\ T_{Air} \end{bmatrix}$. The state without any augmentation is typically included in the set of observable functions. Using the notation from equation 5-10 the terms for $z_{x,k}$ and $z_{xu,k}$ thus become:

$$\mathbf{z}_{x,k} = \begin{bmatrix} T_{Can} \\ T_{Air} \end{bmatrix} \tag{5-12}$$

$$\mathbf{z}_{xu,k} = \begin{bmatrix} H_{\text{CanAir}}(T_{Can}, T_{Air}) \\ H_{\text{PipeAir}}(T_{Pipe}, T_{Air}) \\ R_{GlobSunAir}(u_{ThScr}, d_{iGlob}) \\ H_{\text{AirFlr}}(T_{Flr}, T_{Air}) \\ H_{\text{AirThScr}}(u_{ThScr}, T_{ThScr}, T_{Air}) \\ H_{\text{AirOut}}(T_{Air}, T_{Out}, d_{wind}) \\ H_{\text{AirTop}}(T_{Air}, T_{Top}, u_{ThScr}) \\ H_{\text{GroPipeAir}}(T_{Air}, T_{GroPipe}) \end{bmatrix} \tag{5-13}$$

The term $z_{u,k}$ that refers to observables only dependent on the input u_k is chosen to be simply the input itself. This is because the evolution of the state does not contain nonlinear dynamics that are only dependent on input u_k. Thus, the term for $z_{u,k}$ becomes:

$$\mathbf{z}_{u,k} = \begin{bmatrix} u_{T_{GroPipe}} \\ u_{T_{Pipe}} \\ u_{roof} \\ u_{ThScr} \\ d_{iGlob} \\ d_{T_{out}} \\ d_{wind} \end{bmatrix} \tag{5-14}$$

The combined augmented state is a concatenation of $\mathbf{z}_{x,k}, \mathbf{z}_{xu,k}$ and $\mathbf{z}_{u,k}$. This is expressed by:

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_{x,k} \\ \mathbf{z}_{xu,k} \\ \mathbf{z}_{u,k} \end{bmatrix} \tag{5-15}$$

The approximation of the Koopman operator K, as expressed in equation 5-10 was found using Tikhonov regularization in the minimization problem given by:

$$\min_K \|\mathbf{z}_{x,k+1} - K\mathbf{z}_k\|_F + \lambda^2 \|K\|_F \tag{5-16}$$

The implementation of the KIC model used the same change of notation to calculate the pseudo-inverse as was done for the DMDc algorithm from section 3-2, that was described in [13]. The implemented KIC model was constructed using the algorithm described by 2:

Algorithm 2: KIC algorithm

Input: $X_{0,N}$, $\Upsilon_{0,N}$

Output: K

1 Define a library of lifting functions $\mathbf{g}(x, u)$

2 Apply the lifting functions to the data matrices $\mathbf{Z} = \mathbf{g}(X_{0,N-1}, \Upsilon_{0,N-1})$

3 Compute $V = X_{1,N}\mathbf{Z}^T$

4 Compute $M = \mathbf{Z}\mathbf{Z}^T$

5 Solve the regularized optimization problem as $K = V/(M + \lambda I_n)$

The resulting predictor was applied to the test set. The results when compared to the DMDc where the state was not lifted, can be seen in figure 5-1. Table 5-1 compares the RMSE values of the KIC predictor to the MOESP and DMDc predictors:

	KIC	DMDc	MOESP
RMSE T_{Air}	1.1371	0.8386	0.5873
RMSE T_{Air}	1.3369	0.7569	0.6034

Table 5-1: Comparison of RMSE values of KIC predictor with DMDc and MOESP model on the test set

As can be seen in figure 5-1 and table 5-1, the performance of the KIC predictor is in fact worse than that of the DMDc and MOESP models that are using a non-augmented state. Especially during the night time dynamics, where more control is applied to maintain the temperature inside the greenhouse, the KIC predictor has much larger errors and does not seem to capture the dynamics.

5-3 Redefining Sub-system Boundaries

The observable functions that were previously used were based on functions describing the convective heat exchange of the air temperature T_{Air} using the information available to the defined sub-system. The heat exchanges that determine T_{Can} were left out of the augmented state. Furthermore, some heat exchanges, which are quite significant contributions to the air temperature had to be left out from the set of observable functions because they dependent on terms outside the sub-system boundaries. The most significant heat exchange contributing to the air temperature T_{Air} in the GreenLight model is the convective heat exchange with the top of the green house, denoted by H_{AirTop}. Because the temperature of the top T_{Top} cannot be assumed to be known, the only way to include the dynamics of H_{AirTop}, was to also predict the temperature T_{Top}.

This meant to increase the dimension of the predictor. In order to predict T_{Top}, the convective heat exchanges that dominate the evolution of T_{Top} were also included as lifting functions in the augmented state. The differential equation that describes the evolution of T_{Top} is given by:

$$\mathrm{cap_{Top}}\ \dot{\mathrm{T}}_{\mathrm{Top}} = \mathrm{H_{ThScrTop}} + \mathrm{H_{AirTop}} - \mathrm{H_{TopCov,in}} - \mathrm{H_{TopOut}} \tag{5-17}$$

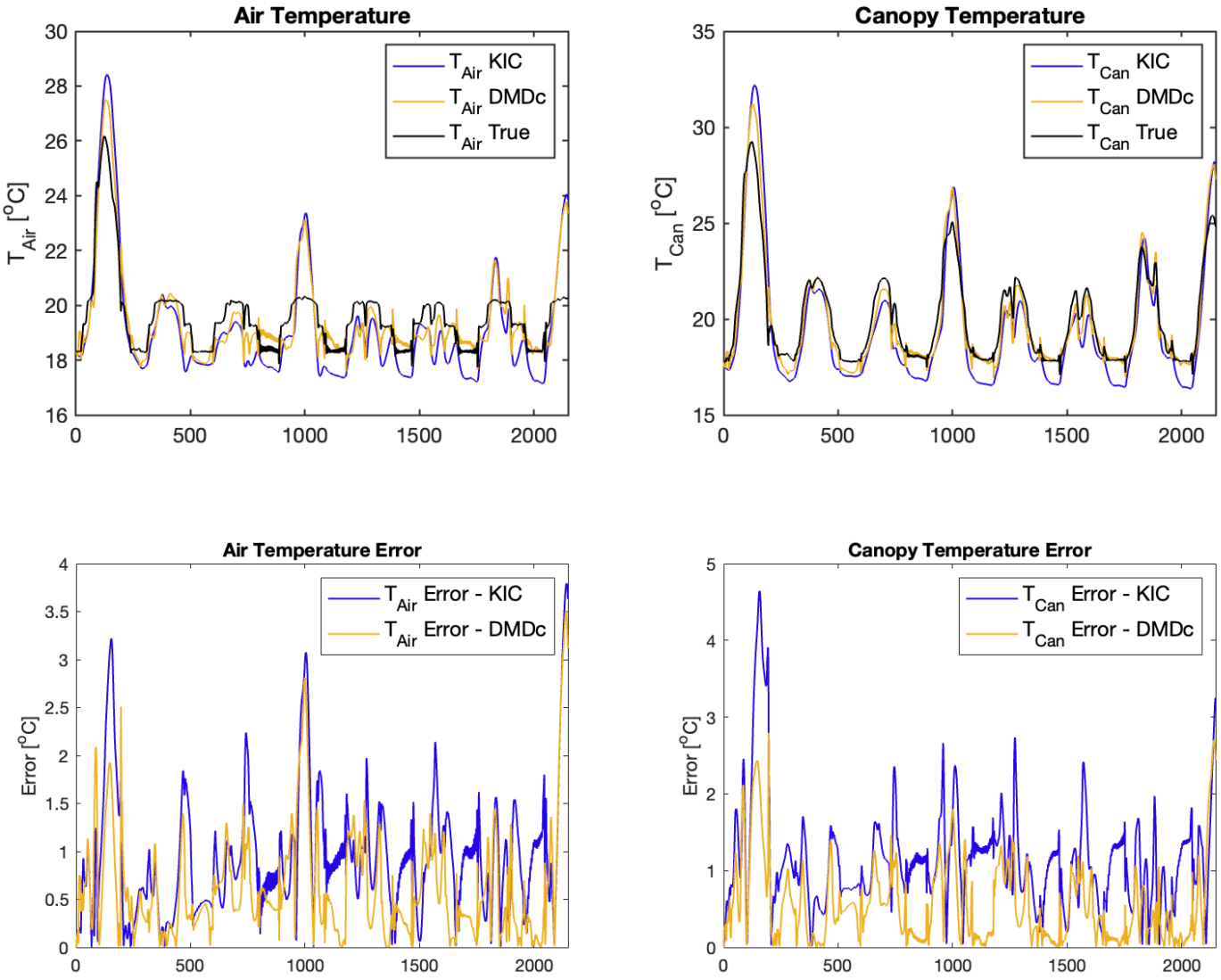

Figure 5-1: KIC and DMDc predictor performance on the test set

While some of these heat exchanges could now be calculated with the expansion of the state, the terms $H_{ThScrTop}$ and $H_{TopCov,in}$ are heat exchanges that could still not be determined.

Thus the temperatures T_{ThScr} and $T_{TopCov,in}$ were also included in the predictor in order to be able to include H_{ThScr} and $H_{TopCov,in}$ as observable functions in the extended state. The differential equations for these terms are given by:

$$\begin{aligned} \mathrm{cap}_{\mathrm{ThScr}}\ \dot{\mathrm{T}}_{\mathrm{ThScr}} =& H_{\mathrm{AirThScr}} + L_{\mathrm{AirThScr}} + R_{\mathrm{CanThScr}} + R_{\mathrm{FlrThScr}} + R_{\mathrm{PipeThScr}} \\ & - H_{\mathrm{ThScrTop}} - R_{\mathrm{ThScrCou,in}} - R_{\mathrm{ThScrSky}} \end{aligned} \tag{5-18}$$

$$\begin{aligned} \mathrm{cap}_{\mathrm{Cov,in}}\ \dot{\mathrm{T}}_{\mathrm{Cov,\ in}} =& H_{\mathrm{TopCov,in}} + L_{\mathrm{TopCov,in}} + R_{\mathrm{CanCov,in}} + R_{\mathrm{FlrCov,\ in}} \\ & + R_{\mathrm{PipeCov,in}} + R_{\mathrm{ThScrCov,\ in}} - H_{\mathrm{Cov,inCov,e}} \end{aligned} \tag{5-19}$$

It can be noted that the augmented state quickly grows if all components of the differential equations are meant to be included. By including T_{Top}, T_{ThScr} and $T_{TopCov,in}$ in the predictions, almost all but five heat exchanges can be included in the extended state. The heat exchanges that remain undetermined in this expansion of the state are: $H_{\mathrm{Cov,inCov,e}}$ $R_{\mathrm{FlrCov,in}}$, $L_{\mathrm{TopCov,in}}$, R_{FlrThScr} and L_{AirThScr} .

Furthermore, the heat exchanges that determine the value for T_{Can}, that are described in equation 2-11, were now also included as observable functions in the augmented state. Only two heat exchanges of equation 2-11 remain undetermined with this expansion of the state. Those are: L_{CanAir} and R_{CanFlr}.

All heat exchanges and their contribution to the dynamics of the temperatures were investigated and the heat exchanges with the floor ($R_{\mathrm{FlrCov,\ in}}$, R_{FlrThScr} , R_{CanFlr}) and the external cover ($H_{\mathrm{Cov,inCov,e}}$) were deemed to be negligible for being warranting an inclusion in the predictors augmented state.

The significant contributions that remained were the heat exchanges due to condensation, which were $L_{\mathrm{TopCov,in}}$, L_{AirThScr} and L_{CanAir}. They depend on the vapour pressure vp_{Air} and vp_{Top} inside the greenhouse, which is also not accessible information for the sub-system configuration described in 2-2. In order to avoid further extension of the augmented state and for the purpose of investigating this approach, the vapour pressure vp_{Air} and vp_{Top} were assumed to be known as external inputs such that the heat exchanges $L_{\mathrm{TopCov,in}}$, L_{AirThScr} and L_{CanAir} could be included in the augmented state.

The new terms for $\mathbf{z}_{x,k}$ and $\mathbf{z}_{u,k}$ remain the same, with the addition of vp_{Air} and vp_{Top} for $\mathbf{z}_{u,k}$. The added heat exchange functions now extend $\mathbf{z}_{xu,k}$ to become:

$$\mathbf{z}_{xu,k} = \begin{bmatrix} H_{CanAir} \\ H_{PipeAir} \\ H_{AirThScr} \\ H_{AirOut} \\ H_{AirTop} \\ H_{GroPipeAir} \\ R_{GlobSunAir} \\ R_{ParSunCan} \\ R_{PipeCan} \\ R_{CanThScr} \\ R_{CanCovIn} \\ H_{ThScrTop} \\ H_{TopCovIn} \\ H_{TopOut} \\ R_{PipeThScr} \\ R_{ThScrCovIn} \\ R_{PipeCovIn} \\ L_{AirThScr} \\ L_{TopCovIn} \end{bmatrix} \tag{5-20}$$

With this expansion, the sub-system now takes on the form as shown in figure 5-2. As can be seen, the extra inputs vp_{Air} and vp_{Top} are assumed to be known in this figure.

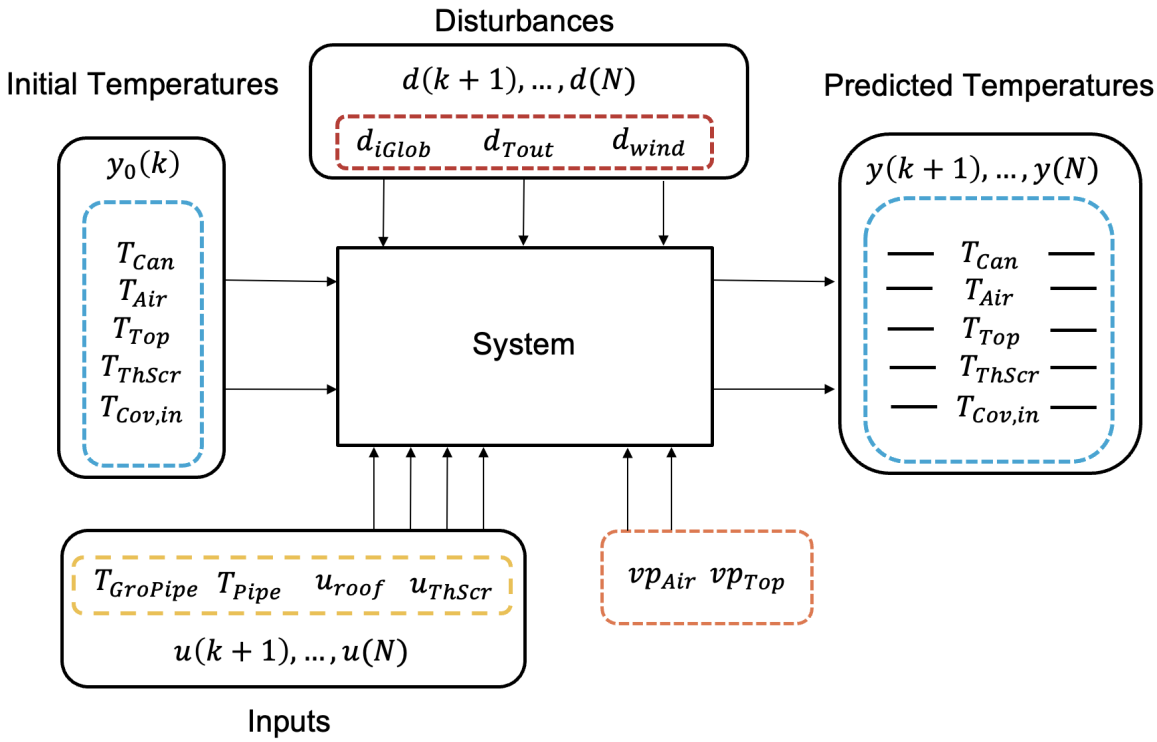

Figure 5-2: Expanded system for investigating KIC approach

The resulting predictions from this expanded system can be seen in figure 5-3. The expanded KIC approach is compared to the DMDc predictor that does not have the augmented state.

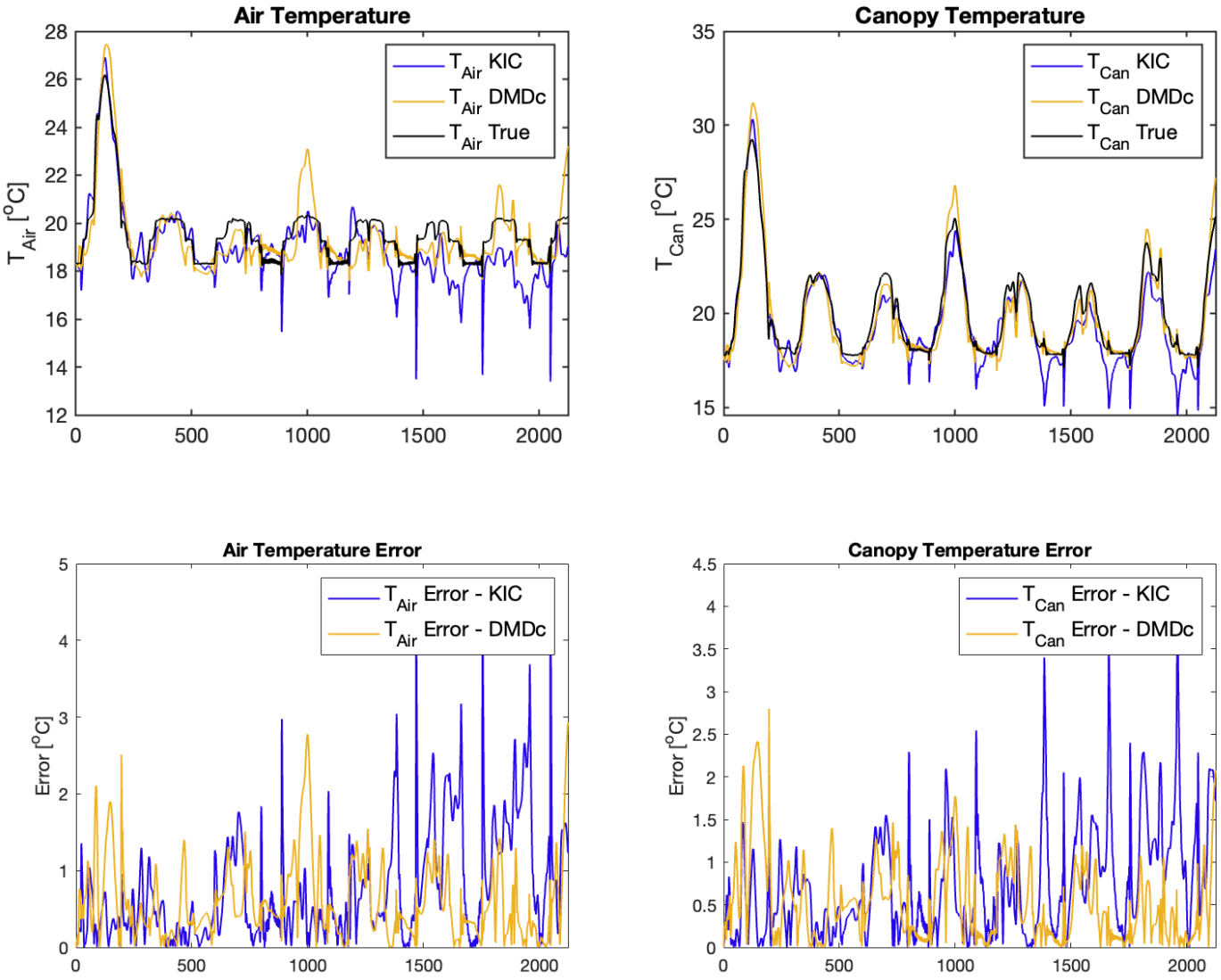

Figure 5-3: Expanded system KIC and DMDc performance on test set

5-4 KIC Problem Analysis

It is noteworthy that on the first day of the expanded system configuration, the KIC predictor seems to have a minor advantage over the DMDc predictor, but over the rest of the test week, it performs a lot worse with an increasing drift of the error. In order to understand why, the performance of the two predictors on a week of the training set was investigated. The two predictors are compared for the air temperature T_{Air} on a week in the training set where little control action was taken. The control inputs in question were the pipe temperature T_{Pipe} and the thermal screen input u_{ThScr} and were plotted over the predictions of the air temperature from both KIC and DMDc predictors. This is shown in figures 5-4 and 5-5, respectively.

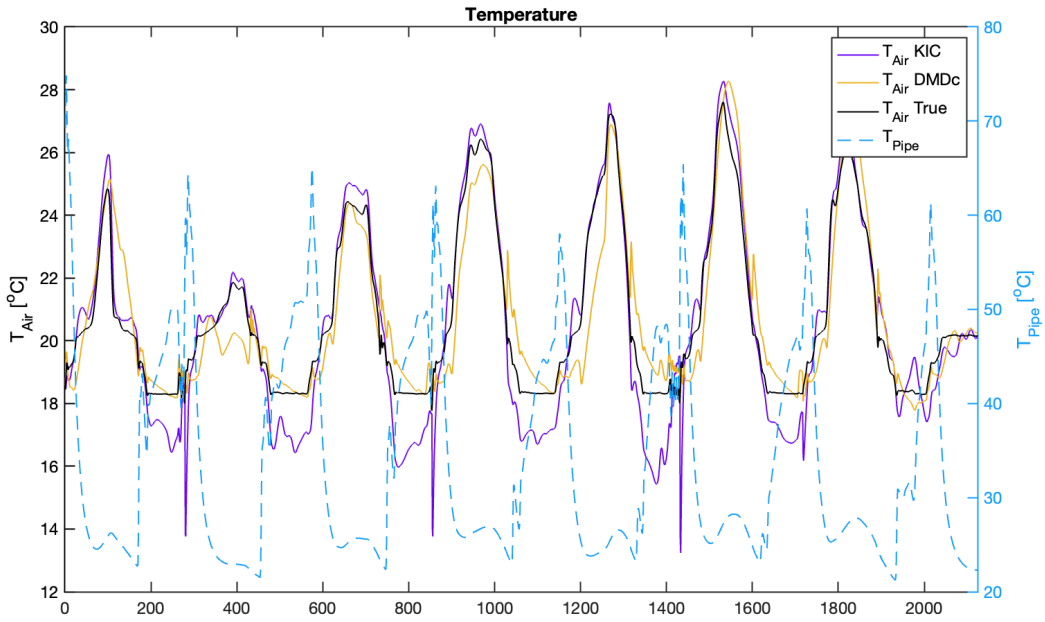

Figure 5-4: Comparison of DMDc and KIC predictor on training set with less T_{Pipe} control action

As can be seen in these figures, the KIC predictor performs better during the day on this week of the training set than the DMDc predictor. Furthermore, it can be noted that the largest errors for the KIC predictor are correlated when more control action is taken. That means when the pipe temperature is much higher than air temperature or when the thermal screens are active. This can be contrasted to a week that is earlier in the training set where more heating occurs and overall more control action is applied to the greenhouse. The control action and the correlated predicted temperatures are shown in figure 5-6 and 5-7.

The most notable difference between 5-6 and 5-4 is that in the former, the more heating occurs also throughout the day. Furthermore, the week depicted in 5-4, shows a heating process with only one peak during the night. In figure 5-6 the heating occurs with more constant control action to keep the heating at a certain level during the night as well as with peaks at the end of the night and the end of the day. It is notable that there is a strong correlation with spikes of the pipe temperature T_{Pipe} and larger errors of the KIC predictor. While there is also some

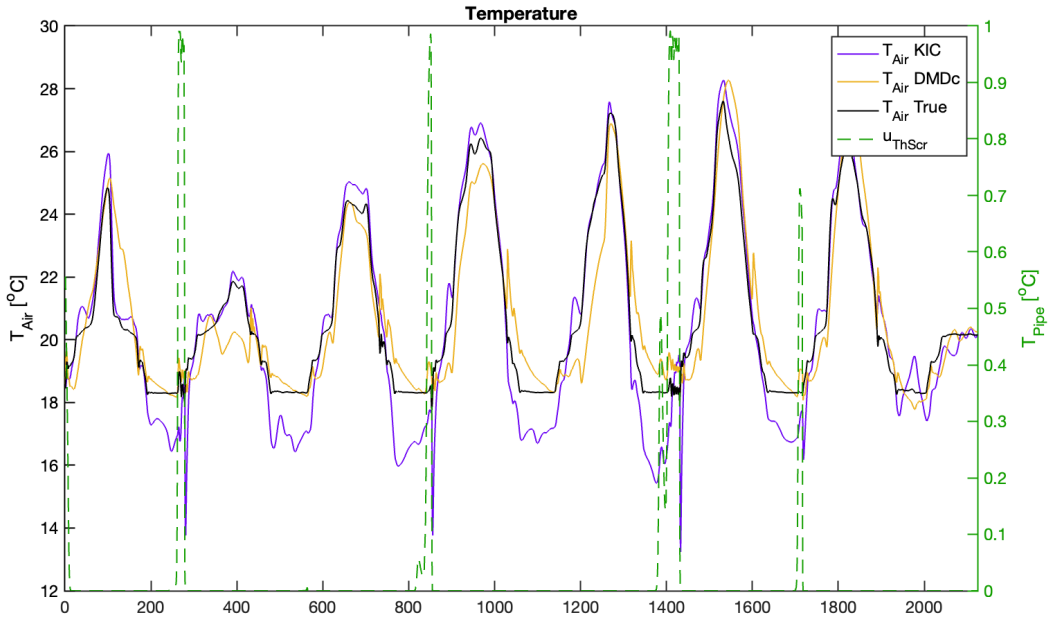

Figure 5-5: Comparison of DMDc and KIC predictor on training set with less u_{thScr} control action

correlation with the thermal screen opening and closing, the thermal screen also is strongly correlated with the heating pipes since they work together to heat the greenhouse. Another notable difference is that there is a much higher frequency in the closing of the thermal screen during the week depicted in 5-7 than in 5-5. The thermal screens in week 5-7 are also closed for longer.

5-4-1 KIC Problem Explanation

The correlation between the error and the heating pipe and thermal screen actuation is likely due to the fact that dynamics related to the heating of the greenhouse remain not included in the augmented state. This explains the difference between the performance of the predictor when there is little heating, where the KIC predictor performs relatively well compared to the DMDc predictor, and the poor performance when a lot of heating actuation occurs. This is also reflected in the later half of the test set depicted in figure 5-3.

A likely explanation is that some heat exchanges that were not included as lifting functions are resulting in the increasing error of the KIC predictor. The heat exchanges with the floor were not included in the set of observable functions due to the fact that those dynamics were deemed negligible. However, the temperature of the floor is also strongly correlated with the heating pipe temperature T_{Pipe}. Thus, excluding the convective heat exchange between the air and the floor temperature, is a possible explanation for why there continues to be large errors when a more temperature control action is applied.

This can be explained from the Koopman theory perspective. From [27], we can state that if the Koopman operator has sufficient eigenfunctions, it might be possible to discover nonlinear

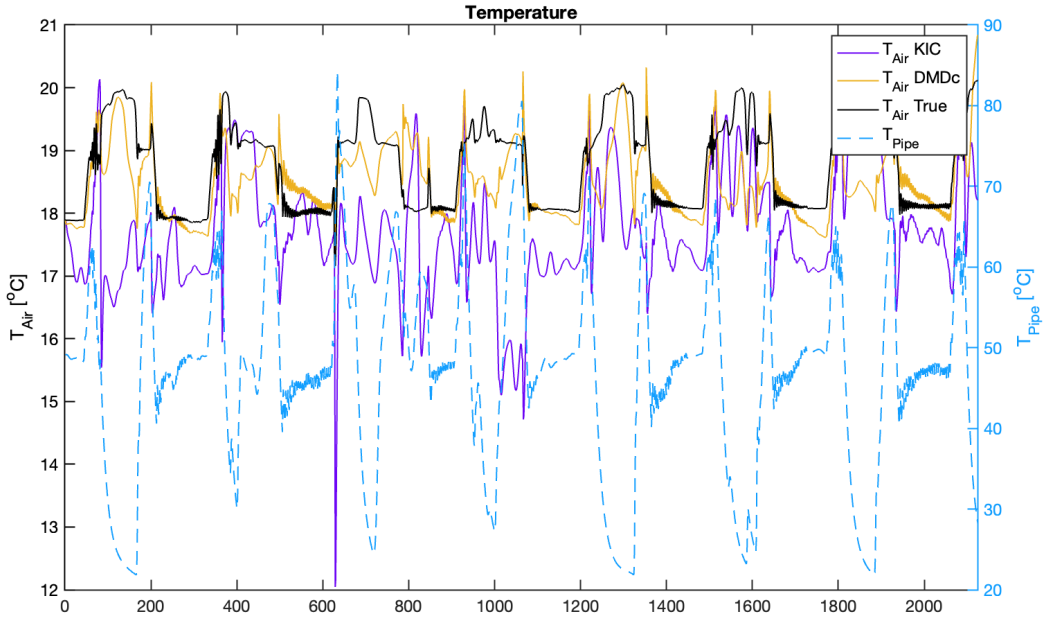

Figure 5-6: Comparison of DMDc and KIC predictor on training set with more T_{Pipe} control action

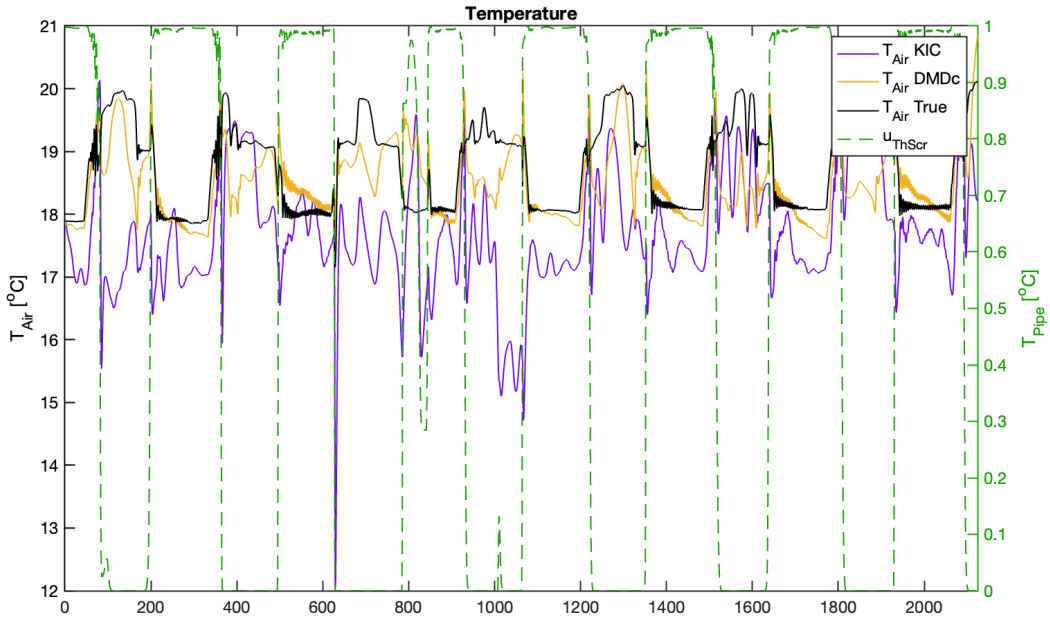

Figure 5-7: Comparison of DMDc and KIC predictor on training set with more u_{thScr} control action

transformation with observable function, where the behavior becomes linear, at least in a specific part of the state space. This transformation can be done using the eigenfunctions of the Koopman operator $\mathcal{K}$. Unfortunately, these eigenfunctions are unknown and infinite dimensions are not practical to work with. The hope is that there exists a finite dimensional subspace S_N that is invariant under $\mathcal{K}$. The projection of $\mathcal{K}$ onto the invariant subspace S_N has been denoted by K. It is also not known if such an invariant subspace exists.

It was hypothesised that the functions that determine the differential equations of the outputs T_{Air} and T_{Can} would span the invariant subspace S_N s.t. $K\mathbf{g}(x) = \mathcal{K}\mathbf{g}(x)$ for any $\mathbf{g}(x) \in S_n$. In most cases, K is an approximation of $\mathcal{K}$ that pointwise converges to $\mathcal{K}$ as $n \to \infty$.

An expansion of the system as illustrated by figure 5-2 was used to include more heat exchanges in the set of observable functions $\mathbf{g}\left(x_k, u_k\right) = \left[g_1\left(x_k, u_k\right), g_2\left(x_k, u_k\right), \ldots g_n\left(x_k, u_k\right)\right]^T$. This increase in dimension n did not yield better results for the entire dynamics of the system.

An explanation for this is that the state x_k and input u_k available to the observable functions $\mathbf{g}\left(x_k, u_k\right)$, was first limited to the outputs and inputs of the sub-system in figure 2-2 and then to the sub-system illustrated by 5-2, while the true dynamics of the simulation also include terms that influence the dynamics of the air temperature that are not included in either sub-system boundaries (such as the floor temperature T_{Flr}).

In short, the set of observables in the first configuration did not span a Koopman invariant subspace S_n and while the expanded system with a larger set of observables showed some improvement for certain dynamics, it still failed to fully span an invariant subspace S_n and was not useful for longer forecasts. However, the correlation of the error with the actuator of the heater that is also strongly correlated with the floor temperature T_{Flr} indicates the source of the error. Furthermore, it seems that the a Koopman invariant subspace S_n is at least partially spanned with these lifting functions, even if that still does not result in a reliable predictor.

A different approach that was not explored is to discover observable functions from the data. One option would be sparse regression of a dictionary of functions like in the Sparse Regression of Nonlinear Dynamics with Control (SINDYc) [4] or learning the observable functions using an artificial neural network as was done by Han et al. (2020) [14].

Finding the correct set of observable functions that will span a Koopman invariant subspace remains the main challenge when applying Koopman theory to real systems. In the expanded system that was presented in this chapter, the invariant subspace could only be partially spanned. The implementations and the necessity to expand the sub-system also illustrate the difficulty of finding the Koopman invariant subspace when the true dynamics are higher dimensional and only part of the state is available. To overcome this issue, the ground truth model should either be reduced or the entire state of the system should be predicted such that the lifted state can include observable functions that include nonlinear dynamics of the full state space.

Chapter 6

Models for Predictive Control

Apart from forecast models for economic predictions and decision making for the grower, another application of prediction models in greenhouses is for model predictive control.

Model predictive control has become has become a very popular control strategy and has seen applications in many industries. It is based on a rolling forecast model between the current time step k and $k+m$, where m is the prediction horizon. The number of future time steps from k to $k+p$ for which optimal control inputs are determined is called the control horizon. This optimal control input calculation is done using a receding horizon as illustrated in figure 6-1.

The first part of this chapter will focus on evaluating the previously derived predictors on their performance for a model predictive control strategy. In the second part, the data enabled predictive control algorithm that is specifically designed for the application in control is presented and implemented and compared to the previously derived models. Lastly, an extension of the data enabled predictive control algorithm is proposed, implemented and compared.

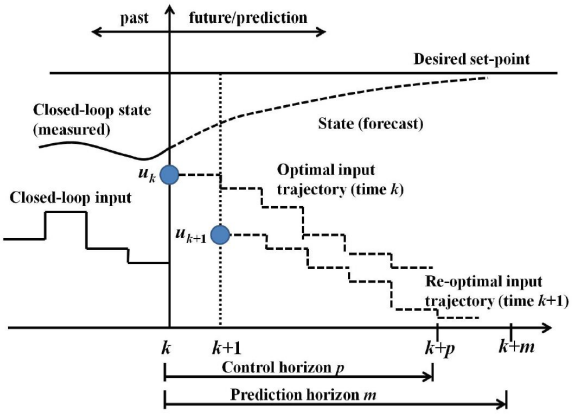

Figure 6-1: Receding Horizon Principle of MPC[10]

6-1 Methodology

Model predictive control implementations use a moving horizon and perform a new prediction of the trajectory at each time-step. To evaluate the different models for their performance in an MPC implementation, the predictions are evaluated on their accuracy. The control action is left out as this is only a comparison of prediction accuracy. The predictions at each time-step are evaluated by calculating the root mean square error (RMSE) between the predicted trajectory and the true trajectory. Since a new prediction is performed at each time-step of the trajectory, the RMSE values of each prediction are then plotted for each time-step. This gives an indication for how accurate the model predicts in an MPC implementation for the chosen prediction horizon. The prediction horizon is set to be 12 time-steps, which is equivalent to one hour.

A schematic of how the performance of the predictors is evaluated is shown in figure 6-2. A lower RMSE value along the trajectory means more accurate predictions in that region and a lower overall RMSE means better performance for an MPC implementation.

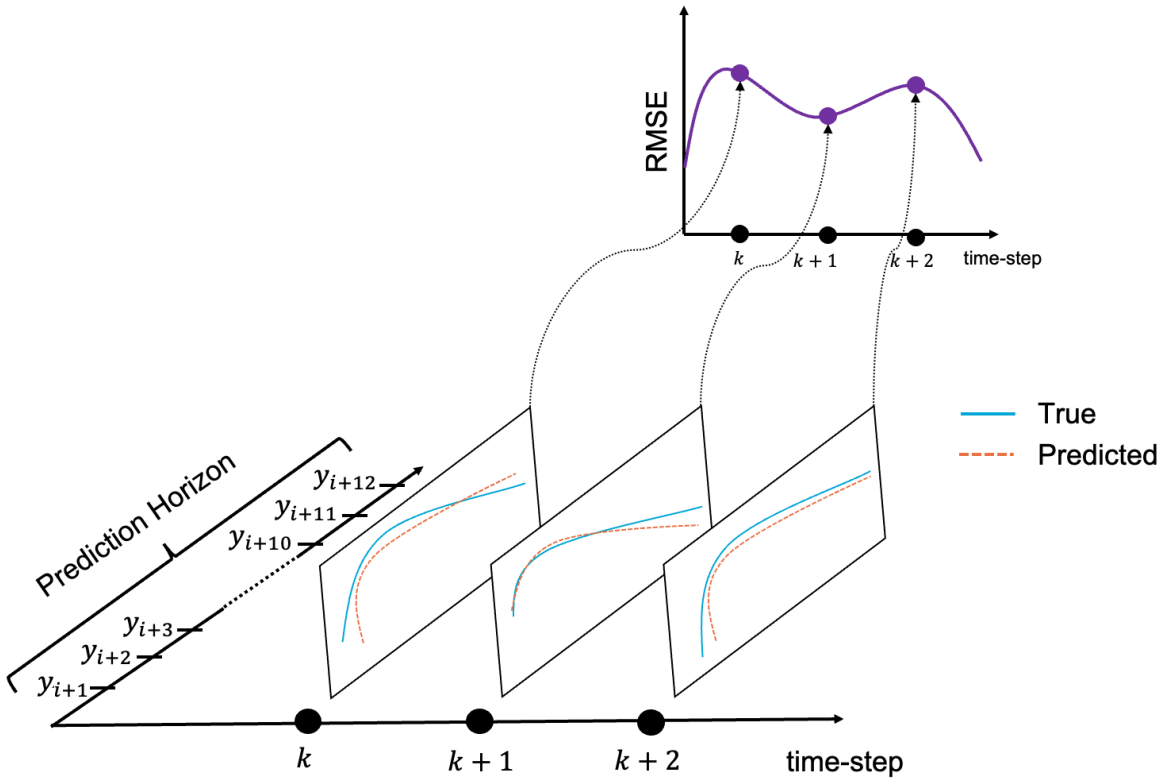

Figure 6-2: Figure illustrating the evaluation of the prediction perfromance for MPC

6-2 LSTM and Linear Models for Predictive Control

The previously implemented Linear Models are reconfigured to perform the predictions according to figure 6-2 and the RMSE of the predictions happening at each time-step is calculated and plotted along the trajectory. The linear models were given a new initial state at

each time-step and the prediction length was changed from previously a week to one hour. The same was done for the stacked LSTM model, except without an initial state since the LSTMs do not make use of initial states. The resulting RMSE values over the first day of the test trajectory can be seen in figure 6-3. As can be seen in this figure, the models with better forecast performance also performed better in this metric, with the ARX and the DMDc models having the highest RMSE values. The MOESP and stacked LSTM model performed best overall. While the stacked LSTM had better performance for the week ahead prediction, there is not a notable difference on this metric that evaluates the performance of one hour ahead predictions.

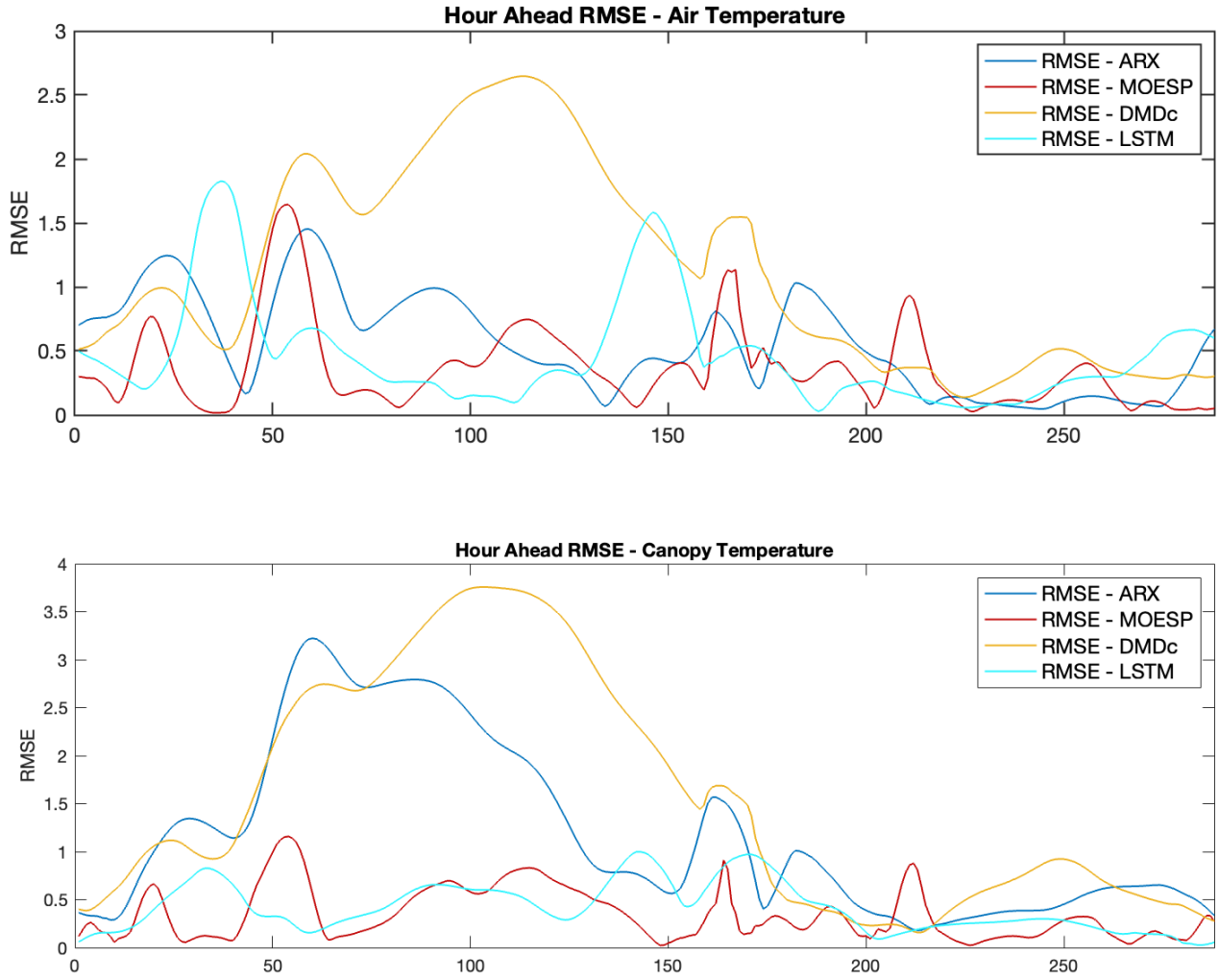

Figure 6-3: Compare Linear Models and stacked LSTM on their moving horizon prediction accuracy with the prediction horizon set to 12 time-steps

6-3 Data-Enabled Predictive Control

A recent approach that has combined behavioural system representation and sequential system identification is the so called Data-Enabled Predictive Control (DeePC) approach proposed by [8]. Its main contribution is the integration of system identification within a model predictive control algorithm. The following section will explain the theory behind the DeePC as well as its implementation and evaluation on its predictive performance for a model predictive control framework. The description of the DeePC algorithm is following the original explaination from Coulso et al. (2018) [8].

6-3-1 Background

Non-Parametric system representation

According to Willem's behavioral theory, the definition of a dynamical system consists of three elements. These elements determine the subspace of the signal space in which the system's trajectories exist. Using this way of describing a system, behavioral system theory has the capability to describe the characteristics of a dynamical system without being reliant on any specific parametric representation of the system. How a dynamical system can be defined by its behaviour alone is given by [8] and is fully described by the following definitions:

Definition 1 Dynamical system is a 3-tuple $(\mathbb{Z}_{>0},\mathbb{W},\mathscr{B})$ where $\mathbb{Z}_{>0}$ is the discrete-time axis, $\mathbb{W}$ is a signal space and $\mathscr{B} \subseteq \mathbb{W}^{\mathbb{Z}_{\geq 0}}$ is the behavior.

Definition 2: Let $(\mathbb{Z}_{>0},\mathbb{W},\mathscr{B})$ be a dynamical system.
(i) $(\mathbb{Z}_{>0},\mathbb{W},\mathscr{B})$ is linear if $\mathbb{W}$ is a vector space and $\mathscr{B}$ is a linear subspace of $\mathbb{W}^{\mathbb{Z}_{\geq 0}}$
(ii) $(\mathbb{Z}_{>0},\mathbb{W},\mathscr{B})$ is time invariant if $\mathscr{B} \subseteq \sigma\mathscr{B}$ where $\sigma : \mathbb{W}^{\mathbb{Z}_{\geq 0}} \to \mathbb{W}^{\mathbb{Z}_{\geq 0}}$ is forward time shift defined by $(\sigma w)(t) = w(t+1)$ and $\sigma\mathscr{B} = \{\sigma w \mid w \in \mathscr{B}\}$.
(iii) $(\mathbb{Z}_{>0},\mathbb{W},\mathscr{B})$ is complete if $\mathscr{B}$ is closed in the topology of pointwise convergence.

When fulfilling condition (i) and (ii), fulfilling condition (iii) is equivalent to $\mathbb{W}$ being finite dimensional. Furthermore, the class of systems $(\mathbb{Z}_{>0}, \mathbb{R}^{n_u+n_y}, \mathscr{B})$ is denoted by $\mathscr{L}^{n_u+n_y}$, where $n_u, n_y \in \mathbb{Z}_{>0}$.

Now that this has been defined, we create a set called:

$$\mathscr{B}_T = \left\{ w \in \left(\mathbb{R}^{n_x+n_u}\right)^T \exists v \in \mathscr{B} s.t. w_t = v_t, \quad 1 \leq t \leq T \right\}.$$

This contains trajectories of length T. It is assumed that the set $\mathscr{B}$ can be broken down into two parts, $\mathscr{B}^u$ and $\mathscr{B}^y$. $\mathscr{B}^u$ represents the input signals, and $\mathscr{B}^y$ represents output signals. They are defined as $\mathscr{B}^u = (\mathbb{R}^m)^{\mathbb{Z}_{\geq 0}}$ and $\mathscr{B}^y \subseteq (\mathbb{R}^p)^{\mathbb{Z}_{\geq 0}}$. Essentially, any trajectory in $\mathscr{B}$ can be represented as a linear combination of input and output signals. This is formulated as any trajectory $w \in \mathscr{B}$ can be expressed as $w = \text{col}(u, y)$, with $\text{col}(u, y) := \left(u^T, y^T\right)^T$. Next, the following two definitions are about two important concepts: controllability and persistency of excitation:

Definition 3: A system is controllable if for every $T \in \mathbb{Z}_{>0}, w^1 \in \mathscr{B}_T, w^2 \in \mathscr{B}$ there exists $w \in \mathscr{B}$ and $T' \in \mathbb{Z}_{>0}$ s.t. $w_t = w_t^1$ for $1 \leq t \leq T$ and $w_t = w_{t-T-T'}^2$ for $t > T + T'$.

This essentially means that a behavioral system is controllable if you can connect any two trajectories in a finite amount of time. Lastly, the definition for persistency of excitation is as follows:

Definition 4: Let $s, T \in \mathbb{Z}_{>0}$ s.t. $T \geq s$. The signal $w = \operatorname{col}(w_1, \ldots, w_T) \in \mathbb{R}^{Tn_u}$ is persistently exciting of order L if the Hankel matrix

$$\mathscr{H}_L(w) := \begin{pmatrix} w_1 & \cdots & w_{T-s+1} \\ \vdots & \ddots & \vdots \\ w_s & \cdots & w_T \end{pmatrix} \tag{6-1}$$

is full row rank.

Parametric system representation

Equivalently, behavioural systems can also be represented in a parametric, state space representation. This is denoted by $\mathscr{B}(A, B, C, D) = \Big\{\operatorname{col}(u, y) \in \left(\mathbb{R}^{n_x+n_u}\right)^{\mathbb{Z}_{\geq 0}} \mid \exists x \in (\mathbb{R}^n)^{\mathbb{Z}_{\geq 0}} \text{ s.t. } \sigma x = Ax + Bu, \quad y = Cx + Du\}$.

The smallest number of parameters for an input/output/state representation, is called a minimal representation. It is referred to with the notation $n(\mathscr{B})$ to refer to its size. Another important system property is lag. This is the smallest positive whole number, denoted as $\ell \in \mathbb{Z}_{>0}$, s.t. the observability matrix described by $\mathcal{O}_\ell(A, C) := \operatorname{col}\left(C, CA, \ldots, CA^{\ell-1}\right)$ has rank $\boldsymbol{n}(\mathscr{B})$. This lag is referred to as $\ell(\mathscr{B})$. Lastly, the lower triangular Toeplitz matrix is denoted by:

$$\mathcal{T}_N(A, B, C, D) := \begin{pmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ CA^{N-2}B & \cdots & CB & D \end{pmatrix} \tag{6-2}$$

With these definitions, uniqueness for the trajectory of the system can be inferred using the following lemmas:

Lemma 6-3.1. *Let $\mathscr{B} \in \mathcal{L}^{n_x+n_u}$ and $\mathscr{B}(A, B, C, D)$ a minimal input/output/state representation. Let $T_{\text{ini}}, N \in \mathbb{Z}_{>0}$ with $T_{\text{ini}} \geq \ell(\mathscr{B})$ and $\operatorname{col}(u_{\text{ini}}, u, y_{\text{ini}}, y) \in \mathscr{B}_{T_{ini}+N}$. Then there exists a unique $x_{\text{ini}} \in \mathbb{R}^{\boldsymbol{n}(\mathscr{B})}$ such that:*

$$y = \mathcal{O}_N(A, C)x_{\text{ini}} + \mathcal{T}_N(A, B, C, D)u \tag{6-3}$$

This result means that if we have enough data about the initial state of a system and the inputs it receives over a significant period, we can uniquely determine the state the system will end up in. Additionally, if we know certain mathematical matrices (A, B, C, D), we can calculate the exact initial state (x_{ini}) of the system.

The following lemma is known as the Fundamental Lemma in behavioural systems theory:

Lemma 6-3.2. *Consider a controllable system $\mathscr{B} \in \mathcal{L}^{n_x+n_u}$. Let $T, s \in \mathbb{Z}_{>0}$, and $w = \text{col}(u, y) \in \mathscr{B}_T$. Assume u to be persistently exciting of order $s+\boldsymbol{n}(\mathscr{B})$. Then* $\text{colspan}\left(\mathscr{H}_s(w)\right) = \mathscr{B}_s$.

What this effectively means is that given input-output dataset $(u[0, T-1], y[0, T-1])$, represented in a Hankel matrix as described by:

$$\left[\begin{array}{c} U_{0,s,T-s+1} \\ \hline Y_{0,s,T-s+1} \end{array}\right] = \left[\begin{array}{cccc} u(0) & u(1) & \dots & u(T-s) \\ u(1) & u(2) & \dots & u(T-s+1) \\ \vdots & \vdots & \ddots & \vdots \\ u(s-1) & u(s) & \dots & u(T-1) \\ \hline y(0) & y(1) & \dots & y(T-s) \\ y(1) & y(2) & \dots & y(T-s+1) \\ \vdots & \vdots & \ddots & \vdots \\ y(s-1) & y(s) & \dots & y(T-1) \end{array}\right] \tag{6-4}$$

And assuming the underlying system can be represented by a Linear Time-Invariant system, is controllable, observable and $u[0, T-1]$ is persistently exciting of order $n_x + s$, then any s-long trajectory can be expressed as:

$$\left[\begin{array}{c} u_{[0,s-1]} \\ y_{[0,s-1]} \end{array}\right] = \left[\begin{array}{c} U_{0,s,T-s+1} \\ Y_{0,s,T-s+1} \end{array}\right] \cdot g \tag{6-5}$$

with $g \in \mathbb{R}^{T-s+1}$. This fundamental lemma is what is utilized for predicting future trajectories. Using a T-long input output trajectory, a linear combination of the columns of the block hankel matrix $\left[\begin{array}{c} U_{0,s,T-s+1} \\ Y_{0,s,T-s+1} \end{array}\right]$ will give an s-long input-output trajectory of the system.

6-3-2 Prediction Model of DeePC

The underlying assumption is that input-output data (u^d, y^d) of an unknown controllable LTI system $\mathscr{B} \in \mathcal{L}^{n_x+n_u}$ with a minimal representation $\mathscr{B}(A, B, C, D)$ is collected. An important aspect to consider is making sure that this data accurately represents the system's behavior. This means that the input sequence needs to be persistently exciting for a duration equal to $T + \mathbf{n}(\mathscr{B})$ to meet Willem's fundamental lemma. For it to be persistently exciting, it also needs to have at least a certain length, which is given by the inequality:

$$T \geq (n_u + 1)\,(s + \mathbf{n}(\mathscr{B})) - 1 \tag{6-6}$$

For the purpose of prediction, the input-output data is partitioned into two halves, which are referred to as past and future. Note that the term future in this context is referring to the second half of the block Hankel matrix and does not contain actual future values that will be predicted. Past and future block Hankel matrices are both created from previously generated data. The depth of past and future block Hankel matrices are denoted as $T_{ini} \in \mathbb{Z}_{>0}$ and $N \in \mathbb{Z}_{>0}$, respectively. The partitioned Hankel matrices are denoted as:

$$\begin{pmatrix} U_p \\ U_f \end{pmatrix} := \mathscr{H}_{T_{\text{ini}}+N}\left(u^d\right), \begin{pmatrix} Y_p \\ Y_f \end{pmatrix} := \mathscr{H}_{T_{\text{ini}}+N}\left(y^d\right) \tag{6-7}$$

Using Lemma 6-3.2 and the partitioned hankel matrices, a trajectory of length $T_{\text{ini}}+N$ for the system $\mathscr{B}_{T_{\text{ini}}+N}$ can be constructed. In fact, a trajectory represented as $\text{col}\,(u_{\text{ini}}, u_N, y_{\text{ini}}, y_N)$ is only part of $\mathscr{B}_{T_{\text{ini}}+N}$ if and only if there is a $g \in \mathbb{R}^{T-T_{\text{ini}}-N+1}$ that satisfies the condition:

$$\begin{pmatrix} U_{\text{p}} \\ Y_{\text{p}} \\ U_{\text{f}} \\ Y_{\text{f}} \end{pmatrix} g = \begin{pmatrix} u_{\text{ini}} \\ y_{\text{ini}} \\ u_N \\ y_N \end{pmatrix} \tag{6-8}$$

In a model predictive control implementation, this linear relationship described by 6-8 is used as a linear constraint in the optimization problem for the optimal control input. However, since the focus here is on prediction only, the algorithm is modified.

The partitioned matrices are leveraged to formulate an optimization problem to first find the value for g by solving:

$$\begin{aligned} \min_{g} \quad & \|u_f - U_f \cdot g\| \\ \text{s.t.} \quad & \\ & \begin{bmatrix} U_p \\ Y_p \end{bmatrix} \cdot g = \begin{bmatrix} u_{\text{ini}} \\ y_{\text{ini}} \end{bmatrix} \end{aligned} \tag{6-9}$$

Subsequently, this g, together with the block Hankel matrix Y_f, is used to make a prediction through the relation $y_N = Y_f \cdot g$. In the next step, the initial values u_{ini} and y_{ini} are shifted by one time-step, a new optimal g is found, and through a new multiplication with Y_f, the next prediction y_N is found.

6-3-3 Implementation with Greenhouse Data

A description of the adapted DeePC algorithm is given by

Algorithm 3: DeePC based predictor

1 Define T,T_{ini}, and N s.t. inequality 6-6 is satisfied
2 Collect previous input output data sequences of length $T + T_{ini} + N$
3 Construct Hankel matrices U_p, Y_p, U_f and Y_f
4 **while** True **do**
5 Define u_{ini}, y_{ini}, and u_N
6 Compute g by solving the optimization problem 6-9
7 Compute and store the predicted output sequence $y_N = Y_f \cdot g$
8 Shift by one time-steps
9 **end while**

The prediction length N was set to 12 in order to be evaluated on the same metric that the other models were evaluated on. The length of T_{ini} was set to 5-steps, through experimentation. This meant that five time-steps were used for the constraint in the optimization problem described by 6-9.

The length T of the input-output signal was set to 4320 time-steps or equivalent to 15 days worth of data. This meant that the training set for this prediction algorithm was less than that of the previously implemented methods. This is because the longer the value of T, the more computationally intensive the problem becomes, where at some point the computational time exceeds the time until the next time-step. Thus only the 15 days of the training set right before the test set were taken.

The linear least squares problem from equation 6-9 was solved using the built-in MATLAB function *lsqlin* using the interior-point algorithm.

The resulting RMSE values for the hour ahead predictions along the first day of the test set can be seen 6-4. Included in the figure are the RMSE plots of the MOESP and LSTM predictor from figure 6-3, since they were the two best performing from the previous comparison.

As can be see in figure 6-4, the DeePC algorithm has mostly lower RMSE values along the first test day than the other two predictors with only a spike for the air temperature RMSE at around time-step 190, which exceeds the RMSE of the LSTM and MOESP. For the canopy temperature predictions, the RMSE values almost entirely fall below the ones of the LSTM and MOESP predictor.

Both MOESP and DeePC make use of block Hankel matrices. The difference lies in the fact that DeePC does not explicitly solve for system matrices (A, B, C, D). Rather, it attempts to find a linear relation g from the block Hankel matrices directly. While the simulation of the trajectories using MOESP uses one initial state, in the DeePC implementation, the T_ini previous outputs are used as a constraint to calculate g. This means that the DeePC algorithm has more initial information around the dynamics close to the prediction, which can explain the improved performance over MOESP.

6-4 Extending Data Enabled Predictive Control

This section will focus on a proposed extension to the Data-Enabled Predictive Control algorithm. This extension seeks to combine the regular DeePC algorithm with the KIC framework. This proposed method will be referred to as extended DeePC or eDeePC.

As was shown in chapter 5, the Koopman with inputs and control (KIC) framework seeks to approximating the Koopman operator of a system by augmenting the state with nonlinear functions. The motivation is to ultimately end up with a predictor that better reflect the underlying non-linear system. Including the benefits of accounting for the non-linear dynamics of the system while making use of the integrated formulation of DeePC for model predictive control, is the purpose of this extension.

First the background behind why this extension seems feasible is discussed.

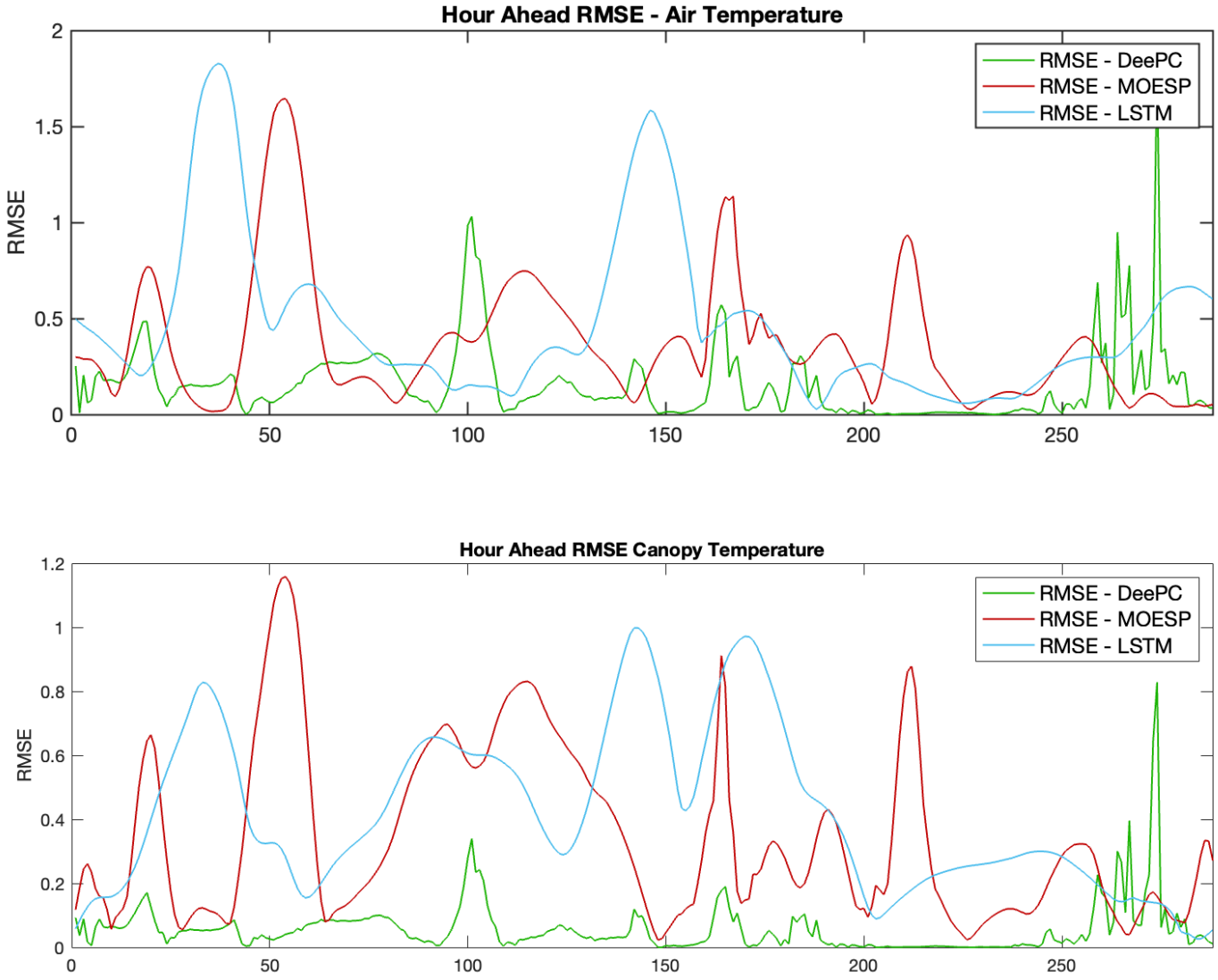

Figure 6-4: Comparison of DeePC, MOESP and stacked LSTM on their moving horizon prediction accuracy with the prediction horizon set to 12 time-steps

6-4-1 Connecting generalized Koopman to DeePC

In the KIC framework described in chapter 5, the correct choice of observable functions $\mathbf{g}(x,u) = [g_1(x,u),\dots,g_m(x,u)]$ can result in a linear system representation for nonlinear dynamics. This equivalent system can be represented as:

$$\mathbf{z}_{x,k+1} = K \begin{bmatrix} \mathbf{z}_{x,k} \\ \mathbf{z}_{xu,k} \\ \mathbf{z}_{u,k} \end{bmatrix} \tag{6-10}$$

Using the same notation as in chapter 5, the obsevables at time step k that are dependent on both input and output are denoted as $\mathbf{z}_{xu,k} = \mathbf{g}_{xu}(x_k,u_k)$. The ones at time-step k that are only dependent on on the input u are denoted as $\mathbf{z}_{u,k} = \mathbf{g}_u(u_k)$ and the ones only dependent on x at k are denoted as $\mathbf{z}_{x,k} = \mathbf{g}_x(x_k)$.

The formulation of equation 6-10 serves to illustrate that in the KIC framework, the evolution of the state occurs in a linear and time invariant manner.

The reason for why this is the case is that the observables can be written as linear combinations of the eigenfunctions of the Koopman operator [29]:

$$\mathbf{g}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} g_1(x,u) \\ g_2(x,u) \\ \vdots \\ g_m(x,u) \end{bmatrix} = \sum_{j=1}^{\infty} \varphi_j(x,u) v_j \tag{6-11}$$

Where the term $\mathbf{v}_j$ is also commonly referred to as the Koopman modes. This can further be expressed with relation to the Koopman operator, since the eigenfunctions span the operator as:

$$\mathcal{K}\mathbf{g}\left(x_k,u_k\right) = \mathbf{g}\left(f\left(x_k,u_k\right)\right) = \sum_{j=1}^{\infty} \lambda_j \varphi_j\left(x_k,u_k\right) v_j \tag{6-12}$$

A finite dimensional approximation of the Koopman operator occurs when the lifting functions span a Koopman invariant subspace S_n. Thus, given set of observables that spans an invariant subspace of the Koopman operator of the underlying system, then the lifted state space will linearly span the output space. In other words, the behaviour of the system in the lifted space will be linear.

6-4-2 Connection to Willem's fundamental Lemma

Willem's fundamental Lemma aims to describe the behaviour of an LTI system $\mathscr{B}(A,B,C,D)$, described by:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned}, \tag{6-13}$$

With Hankel matrices of the input and output dataset $(u_d[0, T-1], y_d[0, T-1])$, as described by equation 6-4. Given that this system is controllable, observable and that the input signal $u_d[0, T-1]$ is persistently exciting of order $n_x + t$, then any s-long trajectory can be expressed as a linear combination of the concatenated Hankel matrix 6-4. The term n_x here refers to the order of the minimal realization of the LTI system.

The proposal of the extended dynamic mode decomposition is to describe the behaviour of the LTI Koopman representation, given by 6-10, through a linear combination of lifted, input-output Hankel matrices.

The input-output data set $(u_d[0, T-1], y_d[0, T-1])$ are transformed with the set of observable functions

$$\mathbf{g}(x, u) = \begin{bmatrix} \mathbf{g}_x(x) \\ \mathbf{g}_{xu}(x, u) \\ \mathbf{g}_u(u) \end{bmatrix} = \begin{bmatrix} z_x \\ z_{xu} \\ z_u \end{bmatrix} \tag{6-14}$$

These new coordinates are used to construct the block Hankel matrices of depth s, using input-output data of length T.

$$\left[\begin{array}{c} Z_{x[0,s,T-s+1]} \\ \hline Z_{xu[0,s,T-s+1]} \\ \hline Z_{u[0,s,T-s+1]} \end{array}\right] = \left[\begin{array}{cccc} z_x(0) & z_x(1) & \dots & z_x(T-s) \\ \vdots & \vdots & \ddots & \vdots \\ z_x(s-1) & z_x(s) & \dots & z_x(T-1) \\ \hline z_{xu}(0) & z_{xu}(1) & \dots & z_{xu}(T-s) \\ \vdots & \vdots & \ddots & \vdots \\ z_{xu}(s-1) & z_{xu}(s) & \dots & z_{xu}(T-1) \\ \hline z_u(0) & z_u(1) & \dots & z_u(T-s) \\ \vdots & \vdots & \ddots & \vdots \\ z_u(s-1) & z_u(s) & \dots & z_u(T-1) \end{array}\right] \tag{6-15}$$

From the linear description of the system in the Koopman invariant subspace it follows that the behaviour $\mathscr{B}(K)$ of the controllable system can be expressed as a linear combination in the lifted space as:

$$\begin{bmatrix} z_{x[0,s-1]} \\ z_{xu[0,s-1]} \\ z_{u[0,s-1]} \end{bmatrix} = \begin{bmatrix} Z_{x[0,s,T-s+1]} \\ Z_{xu[0,s,T-s+1]} \\ Z_{u[0,s,T-s+1]} \end{bmatrix} \cdot g \tag{6-16}$$

That is given the conditions of persistency of excitation and controllability from Lemma 6-3.2 are fulfilled.

6-4-3 Prediction Model of extended DeePC

The same lifting functions for z_x z_{xu} and z_u described in chapter 6-3 by equations 5-12, 5-13 and 5-14 were used for the implementation of the extended DeePC algorithm. Since $z_x = y$ and $z_u = u$ in this formulation, the block Hankel matrices became:

$$\left[\begin{array}{c} Y_{[0,s,T-s+1]} \\ \hline Z_{xu[0,s,T-s+1]} \\ \hline U_{[0,s,T-s+1]} \end{array}\right] = \left[\begin{array}{cccc} y(0) & y(1) & \dots & y(T-s) \\ \vdots & \vdots & \ddots & \vdots \\ y(s-1) & y(s) & \dots & y(T-1) \\ \hline z_{xu}(0) & z_{xu}(1) & \dots & z_{xu}(T-s) \\ \vdots & \vdots & \ddots & \vdots \\ z_{xu}(s-1) & z_{xu}(s) & \dots & z_{xu}(T-1) \\ \hline u(0) & u(1) & \dots & u(T-s) \\ \vdots & \vdots & \ddots & \vdots \\ u(s-1) & u(s) & \dots & u(T-1) \end{array}\right] \tag{6-17}$$

The same approach to predict future outputs as for DeePC was used. The condition for persistency of excitation has to be checked since $\mathbf{n}(\mathscr{B}(K))$ is larger than the minimal realization of the system $\mathscr{B}(A, B, C, D)$ due to increased dimension of the lifted state.

Using the collected data, the block Hankel matrices are divided up into past and future as was done in subsection 6-3-2, now with the addition of a block Hankel matrix that contains the nonlinear transformation using the observable functions:

$$\left(\begin{array}{c} U_p \\ U_f \end{array}\right) := \mathscr{H}_{T_{\text{ini}}+N}\left(u_d\right), \left(\begin{array}{c} Y_p \\ Y_f \end{array}\right) := \mathscr{H}_{T_{\text{ini}}+N}\left(y_d\right), \left(\begin{array}{c} Z_{xu_p} \\ Z_{xu_f} \end{array}\right) := \mathscr{H}_{T_{\text{ini}}+N}\left(\mathbf{g}_{xu}(x,u)\right) \tag{6-18}$$

The partitioned matrices are then also used in the same way as was done in the DeePC approach to first find an optimal g solving the optimization problem given by:

$$\begin{aligned} \min_{g} \quad & \|u_f - U_f \cdot g\| \\ \text{s.t.} \quad & \\ & \left[\begin{array}{c} U_p \\ Z_{xu_p} \\ Y_p \end{array}\right] \cdot g = \left[\begin{array}{c} u_{\text{ini}} \\ z_{xu_{ini}} \\ y_{\text{ini}} \end{array}\right] \end{aligned} \tag{6-19}$$

This g is then used as before with the stacked block Hankel matrix Y_f and Z_{xu_f}, to find the prediction through the relation $\left[\begin{array}{c} y_N \\ z_{xu_N} \end{array}\right] = \left[\begin{array}{c} Y_f \\ Z_{xu_f} \end{array}\right] \cdot g$. From this, the prediction y_N is trivially recovered. Note that the future values of the observables z_{xu_N} are not of interest in this application. In the next step, the initial values u_{ini}, y_{ini} and $z_{xu_{ini}}$ are shifted by one time-step. In the next time-step, a new optimal g is found and with Y_f, the next prediction y_N is found.

It has to be noted that the formulation of the optimization problem 6-19 only works linearly for prediction. For an MPC application, the future values of z_{xu_N} are a free variable in the optimization since it depends on the input u. In DeePC, all matrices are used as linear constraints as described in equation 6-8. Since z_{xu_N} is a result of the nonlinear transformation $z_{xu} = \mathbf{g}_{xu}(x, u)$, this would not function as a linear constraint for the optimization of u as it

would be a nonlinear constraint. Other methods to overcome this challenge for the application in control have to be found.

For the purpose of investigating this approach and to only compare the predictive performance to the regular DeePC, the formulation of the problem in 6-19 works. The implemented algorithm is described as follows:

Algorithm 4: extended DeePC predictor

Define a library of lifting functions $\mathbf{g}(x, u)$
Define T,T_{ini}, and N s.t. inequality 6-6 is satisfied
Collect previous input output data sequences of length $T + T_{ini} + N$
Apply the lifting functions to the input output sequence $z_k = \mathbf{g}(x_k, u_k)$
Construct Hankel matrices U_p, Y_p, U_f, Y_f, Z_{xu_p}, Z_{xu_f}
while True **do**
    Define u_{ini}, y_{ini}, u_N, $z_{xu_{ini}}$
    Compute g by solving the optimization problem 6-19
    Compute and store the predicted output sequence $y_N = Y_f \cdot g$
    Shift by one time-steps
end while

Using the same values as the DeePC with $N = 12$, T_{ini} and $T = 4320$, the resulting comparison can be found in figure 6-5. The length of the input-output data $T = 4320$, still satisfied the condition for persistency of excitation even with increased augment state representation of $\mathscr{B}(K)$. As can be seen in figure 6-5, the regular DeePC approach does perform on average better than the extended DeePC approach. Only at a peak around time-step 180, the extended DeePC has a lower RMSE value and it has much higher peaks at the beginning than the DeePC approach.

6-4-4 Problem explanation for eDeePC

The likely explanation as to why the extended DeePC algorithm did not improve the prediction error over the test day is that as was shown in chapter 5, the set of chosen lifting functions does not span the Koopman invariant subspace S_n. In order to verify this, a day in the training set was also investigated in order to further evaluate the predictor performance. The result can be seen in figure 6-6. As can be seen in this figure, the extended dynamic mode decomposition also performs worse on the training set.

The next step would be to further expand the system as was done in chapter 5 to see if the extended DeePC would have a better performance on a day where the KIC implementation seems to at least partially span a Koopman invariant subspace. However, the problem here would became that if the state was expanded to the formulation of $\mathbf{z}_{xu}$ from equation 5-20, the size of the block Hankel matrix significantly increased to the point that computational time to solve the least squares problem with the linear constrains 6-19 became computationally too expensive.

Furthermore, the individual trajectories were quite well approximated using the linear DeePC algorithm because at the time horizon of 12 time-steps, the evolution of the trajectories

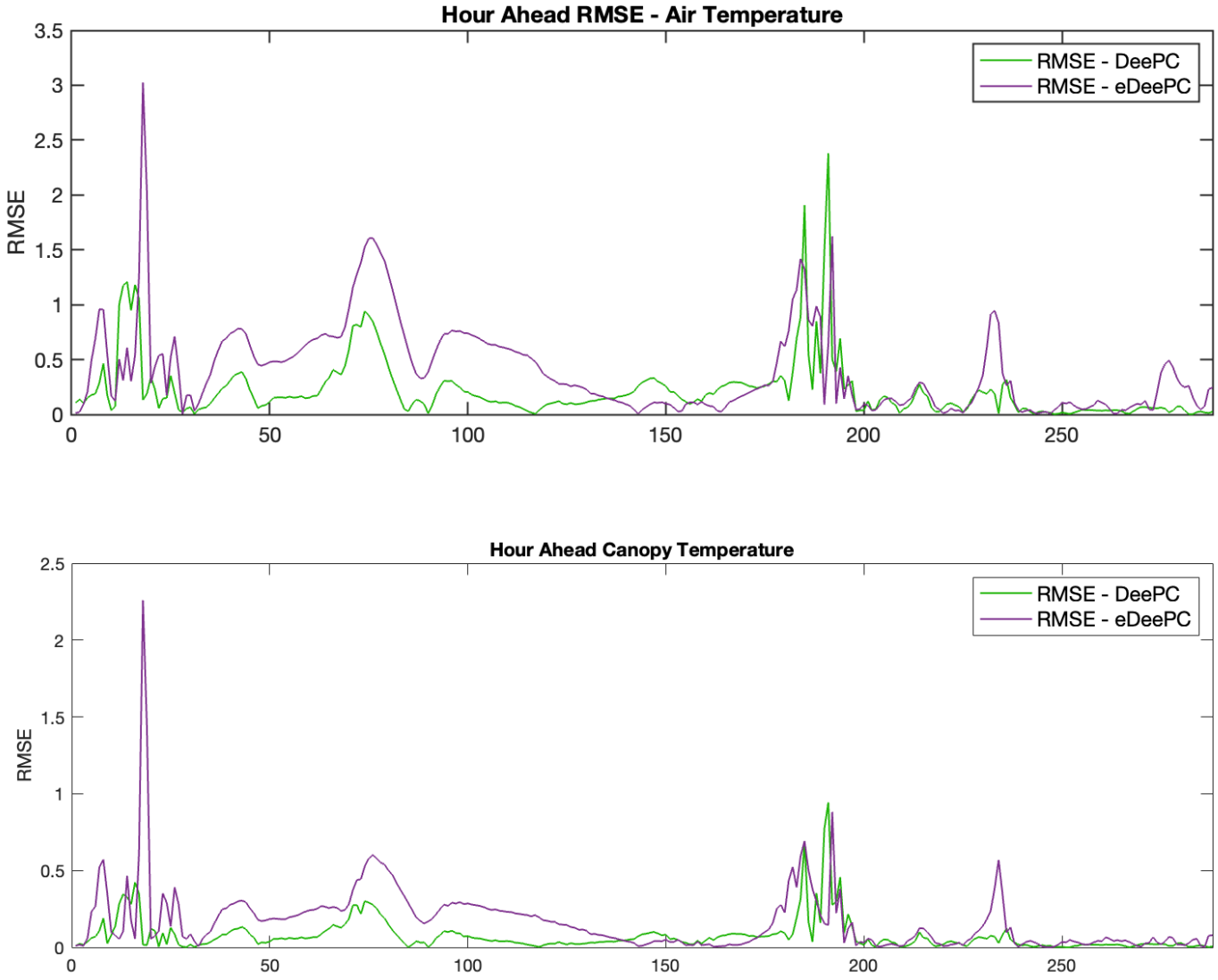

Figure 6-5: Comparison of DeePC, and extended DeePC on their moving horizon prediction accuracy with the prediction horizon set to 12 time-steps on the first day of the test set

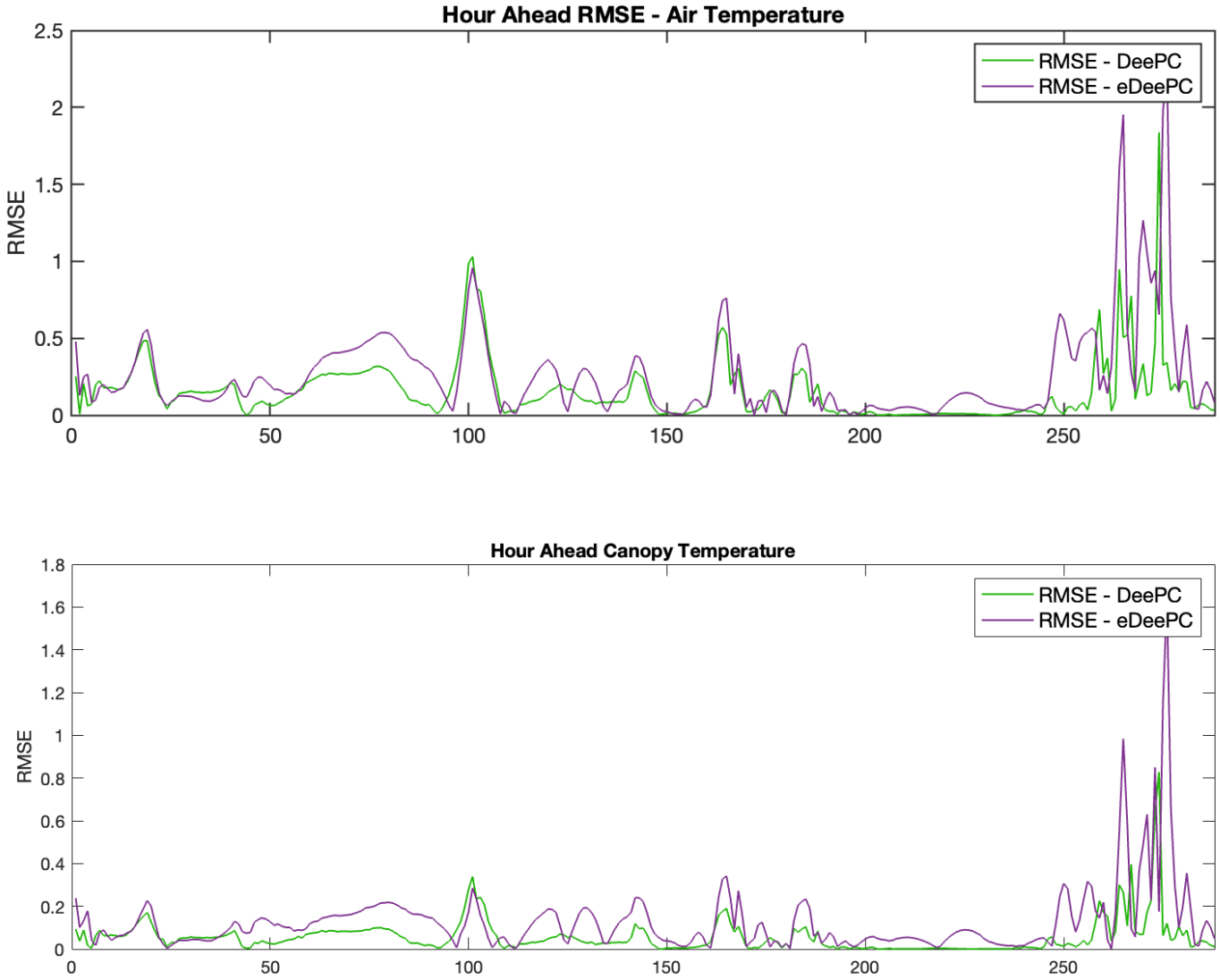

Figure 6-6: Comparison of DeePC, and extended DeePC on their moving horizon prediction accuracy with the prediction horizon set to 12 time-steps on a day during the training set

is close to linear. This becomes clear in the comparison of the MOESP and the LSTM predictor. While the MOESP predictor showed significant improvement in the longer week ahead prediction when compared to the MOESP predictor, it performed very similar on the short time horizon of one hour. Thus, it remains questionable to what degree the nonlinear embedding of the extended DeePC can provide any improvements on the short time horizon of one hour in the application of a greenhouse system.

To evaluate and validate the extended DeePC approach, the algorithm should be tested on a smaller dimensional system where the observable functions of a Koopman invariant subspace are known.

6-5 Discussion

Several forecast models were reconfigured and compared for the short horizon prediction of one hour. It was shown that the nonlinear stacked LSTM model, that showed significant improvements over linear model in the forecast comparison, was performing similarly to the linear models on the shorter time horizon. This is to indicate that nonlinear models do not yield significant performance gains on the shorter time horizon that is typically implemented for a model predictive control strategy in a greenhouse.

Subsequently, the DeePC algorithm was implemented and included in the comparison. It showed an overall improved performance over the stacked LSTM and MOESP models while still having a spikes at certain points of the trajectory. The spikes and and more abrupt changes in prediction accuracy is due to the fact that a least squares problem with linear constraints is performed at every time-step. The other models are identified once and then the same model is used to predict the one hour ahead trajectory at every time-step, which resulted in a smoother trajectory.

The extended DeePC was proposed and implemented. However, it failed to show any improvements over the regular DeePC on both test and training data. This is likely due to the fact that the chosen lifting functions did not fully span the Koopman invariant subspace as was explained in chapter 5. When trying to use the expanded system representation, computational costs became too expensive.

The result from the stacked LSTM further suggests that non-linear models likely provide little benefit for predicting on a one hour time-horizon in the greenhouse climate. Thus the extended DeePC algorithm should be tested on a system where there are gains to be had when accounting for non-linearities.

Chapter 7

Conclusion

7-1 Summary

This thesis project aimed to compare various data driven modelling approaches and evaluate their accuracy on different time horizons. A special focus was set on whether or not Koopman operator based predictors would be able to represent the climate dynamics within a greenhouse. Furthermore, a comparison was made between nonlinear and linear modelling methods for a week long time horizon and a time horizon of one hour.

In the first chapter, the ground truth model that was used to generate the climate data was presented. This generated input-output data was used to create the data driven models in subsequent chapters. The purpose of this was to introduce the reader to the greenhouse system and further illustrate the underlying dynamics to be modelled by the data driven approaches. Furthermore, the sub-system to be approximated was defined, as well as the test and training data sets.

In the next chapter, the linear models were introduced, implemented and compared. The ones included in the comparison were an ARX model, a DMDc model and a subspace identification model based on the MOESP algorithm. These models were compared on the one week prediction horizon and it was found that the MOESP algorithm resulted in the best predictive model for the one week time horizon amongst the linear models.

Chapter 4 introduced LSTM modelling approaches. Different implementations of LSTM models were explored and it was shown that for a prediction horizon of one week, a stacked LSTM architecture would outperform the best linear model forecast. The subsequent chapter went over the Koopman based predictor implementation, which was the other non-linear modelling approach included in this comparison. It was shown that for the chosen sub-system, using lifting functions based on the underlying differential equations of the simulation did not result in a Koopman invariant subspace and did not improve predictions over the DMDc predictor that did not use a lifted state. After expanding the sub-system to be able to include more nonlinear dynamics in the lifted state, the Koopman based predictor showed improvements over the DMDc predictor on certain sections of the training set. These were

sections with less control actuation from the heating pipe and thermal screen. The heat exchange dynamics closely correlated heating pipe temperature remained excluded from the lifted state due to the choice of the sub-system borders. It was concluded that this correlation between control actuation and accuracy of the Koopman based predictor likely meant a partial spanning of the Koopman invariant subspace with the error resulting from the dynamics excluded from the lifted state.

Lastly, the prediction models were also evaluated on a one hour time horizon with the application of control in mind. Of the previously implemented prediction models, it was shown that the advantage of the stacked LSTM model over the one week time horizon diminished on the one hour time horizon, indicating that for the purpose of predictive control models, there seems to not be a significant advantage of using a non-linear LSTM model over linear methods. Furthermore, the DeePC algorithm was adapted to be included in the comparison to evaluate its predictive performance as an algorithm specifically designed for model predictive control. The DeePC algorithm outperformed both the stacked LSTM as well as the MOESP algorithm on the test day. Additionally, an extension to the DeePC algorithm was proposed. The extension proposed to include lifted states as was done in the Koopman operator based predictor with the idea to improve predictions embedding the relevant nonlinear dynamics in the Hankel matrices. This extension, however, did not result in improved predictions, likely cause the choice of lifting functions did not span a Koopman invariant subspace as was previously shown.

7-2 Answer to the Research Question

The main research question was posed as:

How do different data-driven modelling approaches compare as forecast and predictive control models when applied to greenhouse data?

In order to answer this question, the proposed sub-questions were answered.

- *How does the selection of linear models compare to the selection of nonlinear models for the purpose of week ahead forecasts?*

  Of the linear model, the one with the most parameters using the MOESP algorithm performed the best on the week ahead forecast test set. However, all linear models performed worse than the LSTM implementations. Of the LSTM implementations, the stacked architecture performed the best, by being able to encode more of the longer term dynamics in its weights. The KIC implementation had several issues and did not provide any significant improvement over the non-lifted DMDc implementation. Even when the boundary of the sub-system was expanded, the predictor did not improve performance significantly on the test set and only showed minor improvements on some dynamics of the training data where little control action was applied.

- *How can a predictor using the Koopman in control (KIC) framework be implemented and what are its challenges?*

Several different lifting functions were applied to find an approximation of a Koopman invariant subspace. The initial choice of lifting functions did not yield any improvement over the non-lifted DMDc implementation. In fact, the performance of the predictor was worsened through the inclusion of the lifting functions. When the system was expanded and a larger set of lifting functions was chosen, the predictor showed improvement on certain dynamics on the training set, where little control action was applied. However, the main issue remained that the set of lifting functions did not fully span the Koopman invariant subspace S_n. The excluded dynamics of the floor temperature T_{Flr} are a likely source of the error and a possible reason why the Koopman invariant subspace was not fully spanned. The problem became that to further increase the dimension of the lifting functions required to continually expand the borders of the sub-system. This defeated the purpose of choosing to model a sub-system in the first place. The approach to choose the lifting functions based on the equations of the simulation when only partial state and input information was available limited the ability to increase the dimension of the observable functions. However, previous attempts that used common choices of lifting functions had not resulted in any acceptable results.

- *What data-driven modelling approaches included in the comparison are best suited for model predictive control in the greenhouse system?*

  All of the models that were initially implemented as forecast models were reconfigured to be evaluated on the shorter time horizon of one hour. Here it became clear that, while the LSTM architectures had advantages on the longer forecast, its benefits on the shorter time horizon diminished and its performance was in the same order as the MOESP subspace identification approach. The DeePC algorithm had the best performance on the test day, scoring the lowest RMSE values. It is an algorithm, specifically designed for short term predictions with the application of predictive control in mind and integrates easily with an objective function of a model predictive control algorithm. By implicitly finding the behaviour of the system for the next few time-steps through constraints using multiple previous outputs together with block Hankel matrices, the DeePC algorithm achieves better performance than any of the other predictors. The proposed extended DeePC (eDeePC) algorithm did not yield any improvements on the test set over the DeePC algorithm.

- *How can a Koopman theory be included in the formulation of the DeePC algorithm and how does it perform?*

  Embedding nonlinear dynamics in the Hankel matrices of the DeePC algorithm was a proposed extension to include Koopman theory. Using the same lifting functions of the initial KIC implementation did not yield any improvements on the test or training set. Implementing the expanded system using a larger set of lifting functions significantly increased the computational time to the point that computation became unfeasible. In sight of the results of the LSTM predictors performance on the short time horizon, nonlinear embeddings in the block Hankel matrices might not provide significant improvement on the time horizon of one hour for the application of greenhouse climate prediction.

7-3 Recommendations for Future Work

LSTM Modelling: The stacked LSTM model showed the best results amongst the implemented LSTM architectures and the trend in this work has been that a larger number of parameters increases prediction accuracy. Therefore it is recommended to further investigate how different kind of additional layers affect the results. Moreover, increasing the order of the LSTM together with a dense layer for the output might enhance performance. Furthermore, to fulfill its actual purpose, the stacked LSTM could be integrated with an economic forecast model to aid decision making of the grower as has been done with single layer LSTMs. For the application in control, the stacked LSTM does not show significant promise over the DeePC approach and with the additional difficulty of the non-linearities in the LSTM for optimization, there seems little reason to investigate LSTM models with the purpose of control in greenhouses.

KIC Modelling: The implementation of the KIC approach encountered issues since the chosen observable functions did not entirely span a Koopman invariant subspace. For future work this means to either try to find these functions from the data. One promising approach for finding the right lifting functions from data is using Sparse Identification of Nonlinear Dynamics with Control (SINDYc) [4]. There also have been approaches for learning the right choice of observable functions using deep neural networks [42]. Another route to take would be to try to use the same approach as was done in this work but to have the ground truth data be coming from a smaller model. If the underlying model is smaller, it would be possible to represent all heat exchange functions in the lifted state. An alternative to this route would be to attempt to predict the entire state of the GreenLight model in order to be able to include all heat exchange functions in the lifted state. However, for real world applications, data-driven discovery of the obsevables should be prioritized since it is not guaranteed that custom observable functions from a simulation data will also result in a good predictor for real world applications.

Extended Data Enabled Predictive Control: The implementation of the extended data enabled predictive control algorithm could not be fully verified. In order to further investigate this method, the algorithm should be implemented in a smaller system where the observable functions that span a Koopman invariant subspace are fully known, such as the van der Pol Oscillator. For the application in the greenhouse system, there only seems to be marginal gains to be had. Additionally, there are issues caused by dimensionality that affect the computational time when the lifted state becomes large. The most promising application of this algorithm is hence in systems where non-linearities strongly affect the short term behaviour and where the set of observables does not drastically increase the dimension of the lifted state. As it was presented, this method did not provide an explanation for how the optimization for a control input sequence would be performed when a constraint would be dependent on a nonlinear observable function. This warrants further investigation, where several approaches could be explored.

Bibliography

[1] Mukand S. Babel, Victor R. Shinde, Devesh Sharma, and Nguyen Mai Dang. Measuring water security: A vital step for climate change adaptation. *Environmental Research*, 185, June 2020. doi:10.1016/j.envres.2020.109400.

[2] Kurt Benke and Bruce Tomkins. Future food-production systems: vertical farming and controlled-environment agriculture. *Sustainability: Science, Practice and Policy*, 13:13–26, January 2017. doi:10.1080/15487733.2017.1394054.

[3] Daniel Bruder, Brent Gillespie, C. David Remy, and Ram Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. *arXiv preprint arXiv:1902.02827*, February 2019. URL: http://arxiv.org/abs/1902.02827.

[4] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49:710–715, 2016. doi:10.1016/j.ifacol.2016.10.249.

[5] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4), December 2012. URL: http://dx.doi.org/10.1063/1.4772195, doi:10.1063/1.4772195.

[6] Nicholas Cartocci, Agnese Monarca, Gabriele Costante, Mario Luca Fravolini, K. Merve Dogan, and Tansel Yucelen. Linear control of a nonlinear aerospace system via extended dynamic mode decomposition. *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*, 2022. doi:10.2514/6.2022-2046.

[7] Liette Connolly-Boutin and Barry Smit. Climate change, food security, and livelihoods in sub-saharan africa. *Regional Environmental Change*, 16:385–399, February 2016. doi:10.1007/s10113-015-0761-x.

[8] Jeremy Coulson, John Lygeros, and Florian Dörfler. Data-enabled predictive control: In the shallows of the deepc. In *18th European Control Conference (ECC)*, pages 307–312. IEEE, November 2019. URL: http://arxiv.org/abs/1811.05890.

[9] Steven Dahdah and James R. Forbes. System norm regularization methods for koopman operator approximation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478, September 2022. doi:10.1098/rspa.2022.0162.

[10] Li Dai, Yuanqing Xia, Mengyin Fu, and M Mahmoud. Discrete-time model predictive control. In *Advances in Discrete Time Systems*, pages 77–116. Rijeka: IntechOpen, 2012. doi:10.5772/3432.

[11] Hendrik Feije de Zwart. *Analyzing energy-saving options in greenhouse cultivation using a simulation model*. Landbouwuniversiteit Wageningen, Netherlands, April 1996.

[12] Michael Deflorian and Susanne Zaglauer. Design of experiments for nonlinear dynamic system identification. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 44:13179–13184, 2011. doi:10.3182/20110828-6-IT-1002.01502.

[13] Camilo Garcia-Tenorio and Alain Vande Wouwer. A matlab toolbox for extended dynamic mode decomposition based on orthogonal polynomials and p-q quasi-norm order reduction. *Mathematics*, 10, October 2022. doi:10.3390/math10203859.

[14] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. *Proceedings of the IEEE Conference on Decision and Control*, pages 1890–1895, December 2020. doi:10.1109/CDC42340.2020.9304238.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, November 1997. doi:10.1162/neco.1997.9.8.1735.

[16] E. Iddio, L. Wang, Y. Thomas, G. McMorrow, and A. Denzer. Energy efficient operation and modeling for greenhouses: A literature review. *Renewable and Sustainable Energy Reviews*, 117, January 2020. doi:10.1016/j.rser.2019.109480.

[17] Dae Hyun Jung, Hyoung Seok Kim, Changho Jhin, Hak Jin Kim, and Soo Hyun Park. Time-serial analysis of deep neural network models for prediction of climatic conditions inside a greenhouse. *Computers and Electronics in Agriculture*, 173, June 2020. doi:10.1016/j.compag.2020.105402.

[18] Eurika Kaiser, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2, September 2021. doi:10.1088/2632-2153/abf0f5.

[19] David Katzin, Simon van Mourik, Frank Kempkes, and Eldert J. van Henten. Greenlight – an open source model for greenhouses with supplemental lighting: Evaluation of heat requirements under led and hps lamps. *Biosystems Engineering*, 194:61–81, June 2020. doi:10.1016/j.biosystemseng.2020.03.010.

[20] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[21] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, July 2018. doi:10.1016/j.automatica.2018.03.046.

[22] J. Nathan Kutz, J. L. Proctor, and S. L. Brunton. Applied koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems. *Complexity*, 2018:1–16, 2018. doi:10.1155/2018/6010634.

[23] Yuwen Liu, Dejuan Li, Shaohua Wan, Fan Wang, Wanchun Dou, Xiaolong Xu, Shancang Li, Rui Ma, and Lianyong Qi. A long short-term memory-based model for greenhouse climate prediction. *International Journal of Intelligent Systems*, 37:135–151, January 2022. doi:10.1002/int.22620.

[24] Alexandre Mauroy, Igor Mezić, and Yoshihiko Susuki. *The Koopman Operator in Systems and Control Concepts, Methodologies, and Applications: Concepts, Methodologies, and Applications*. Springer, January 2020. doi:10.1007/978-3-030-35713-9.

[25] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, August 2005. doi:10.1007/s11071-005-2824-x.

[26] Thomas Erik Mulder, Sven Baars, Fred W. Wubs, Inti Pelupessy, Merijn Verstraaten, and Henk A. Dijkstra. Symbiotic ocean modeling using physics-controlled echo state networks. *Journal of Advances in Modeling Earth Systems*, April 2022. doi:10.1002/essoar.10511109.1.

[27] Samuel E Otto and Clarence W Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:59–87, May 2021. doi:10.1146/annurev-control-071020.

[28] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, September 2014. URL: http://arxiv.org/abs/1409.6358, doi:10.1137/15M1013857.

[29] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Generalizing koopman theory to allow for inputs and control. *SIAM Journal on Applied Dynamical Systems*, 17:909–930, 2018. doi:10.1137/16M1062296.

[30] Fathi M. Salem. *Recurrent neural networks: From simple to gated architectures*. Springer International Publishing, January 2022. doi:10.1007/978-3-030-89929-5.

[31] Abel Sancarlos, Morgan Cameron, Jean Marc Le Peuvedic, Juliette Groulier, Jean Louis Duval, Elias Cueto, and Francisco Chinesta. Learning stable reduced-order models for hybrid twins. *Data-Centric Engineering*, 2, August 2021. doi:10.1017/dce.2021.16.

[32] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010. doi:10.1017/S0022112010001217.

[33] Yongtao Shen, Ruihua Wei, and Lihong Xu. Energy consumption prediction of a greenhouse and optimization of daily average temperature. *Energies*, 11(1), January 2018. doi:10.3390/en11010065.

[34] Dirk van Bokkem, Max van den Hemel, Sebastijan Dumančić, and Neil Yorke-Smith. Embedding a long short-term memory network in a constraint programming framework

for tomato greenhouse optimisation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(13):15731–15737, September 2023. URL: https://ojs.aaai.org/index.php/AAAI/article/view/26867, doi:10.1609/aaai.v37i13.26867.

[35] B.H.E. Vanthoor, C. Stanghellini, E.J. van Henten, and P.H.B. de Visser. A methodology for model-based greenhouse design: Part 1, a greenhouse climate model for a broad range of designs and climates. *Biosystems Engineering*, 110(4):363–377, 2011. URL: https://www.sciencedirect.com/science/article/pii/S1537511011000948, doi:10.1016/j.biosystemseng.2011.06.001.

[36] Michel Verhaegen and Vincent Verdult. *Filtering and system identification : a least squares approach*. Cambridge University Press, 2007.

[37] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, December 2015. doi:10.1007/s00332-015-9258-5.

[38] Sylvan H. Wittwer and Nicolas Castilla. Protected cultivation of horticultural crops worldwide. *HortTechnology horttech*, 5(1):6–23, 1995. URL: https://journals.ashs.org/horttech/view/journals/horttech/5/1/article-p6.xml, doi:10.21273/HORTTECH.5.1.6.

[39] Frank Xiao. Time series forecasting with stacked long short-term memory networks. *arXiv preprint*, November 2020. URL: http://arxiv.org/abs/2011.00697, doi:10.48550/arXiv.2011.00697.

[40] Huihui Yu, Yingyi Chen, Shahbaz Gul Hassan, and Daoliang Li. Prediction of the temperature in a chinese solar greenhouse based on lssvm optimized by improved pso. *Computers and Electronics in Agriculture*, 122:94–102, March 2016. URL: https://www.sciencedirect.com/science/article/pii/S0168169916000247, doi:10.1016/j.compag.2016.01.019.

[41] Vrushabh Zinage and Efstathios Bakolas. Koopman operator based modeling for quadrotor control on se(3). *IEEE Control Systems Letters*, 6:752–757, March 2022. URL: http://arxiv.org/abs/2103.03363, doi:10.1109/LCSYS.2021.3085963.

[42] Z. Zuo, H. Mao, X. Zhang, J. Hu, L. Han, and J. Ni. Forecast model of greenhouse temperature based on time series method. *Nongye Jixie Xuebao/Transactions of the Chinese Society of Agricultural Machinery*, 41(11):173–182, 2010. doi:10.3969/j.issn.1000-1298.2010.11.034.