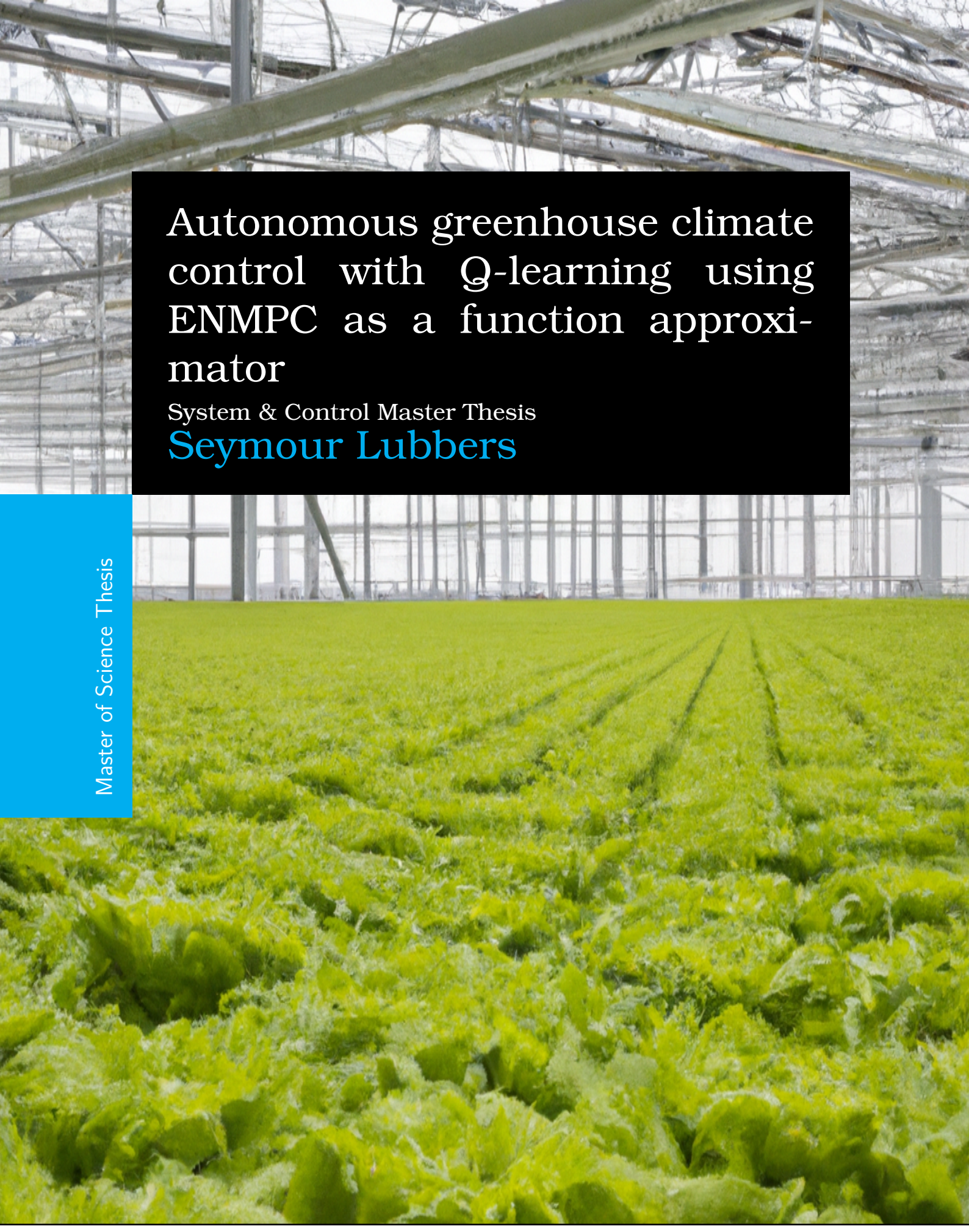

Autonomous greenhouse climate control with Q-learning using ENMPC as a function approximator

System & Control Master Thesis

Seymour Lubbers

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Autonomous greenhouse climate control with Q-learning using ENMPC as a function approximator

System & Control Master Thesis

Master of Science Thesis

Seymour Lubbers

August 6, 2023

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

Abstract

Greenhouses allow production of crops that would otherwise be impossible. Permitting more local, fresher and nutrient richer crop production. Efforts are taken to minimize societal harm due to energy and resource consumption by greenhouse production systems. One way to control such systems is by using model predictive control. Optimal crop yield and resource efficiency can, in theory, be achieved by model predictive control. Unfortunately, one major drawback of model predictive control is that it is not well equipped to deal with parametric uncertainty. Significant prediction errors can occur when a mismatch between the model and the real system exists, resulting in deteriorated performance of the system. Strategies exist, such as robust MPC, that are designed to handle uncertainty, but those often result in conservative control policies. This thesis proposes to use model predictive control as a function approximator for RL in order to learn values for model and MPC parameters that can deliver optimal performance in the case of model mismatch.

In this thesis, data-driven economic nonlinear model predictive control using Q-learning is proposed as a method to alter the model parameters. The performance of the system after learning is compared to approaches using robust and nominal model predictive control. Three different goals are determined: maximizing economic profit, minimizing the constraint violations and maximizing the economic performance while minimizing constraint violations.

In this work, an ENMPC scheme is used as a function approximator in a Q-learning environment. The optimization solution from the ENMPC scheme is used as the input to the system, while the Q-learning agent optimizes the parameter values of the ENMPC scheme and model for the environment. The performance of the system after learning is compared to approaches using robust and nominal model predictive control. The simulation results show that the data-driven ENMPC using reinforcement learning is able to decrease constraint violations by up to 94%, but unable to increase economic performance compared to nominal MPC, compared to robust MPC the EPI is increased by almost 10% while keeping constraint violations at a similar level.

Table of Contents

Acknowledgements

This thesis has been a labor of perseverance and determination. Overcoming some time-consuming setbacks and unsatisfying results, the result is finally here, and I would like to thank some people who have helped me along the way.

First of all, I would like to thank my supervisor Azita Dabiri for her assistance during the writing of this thesis, for asking critical questions during our meetings and providing guidance when I was looking for direction.

Thanks to Congcong Sun for supervising me from Wageningen University and Research, for helping me with all questions related to the field of greenhouse climate control and understanding the field of research.

Filippo and Bart are deserving of a mention of their own, being critical in providing me with ideas and input for the Control and greenhouse sections of this thesis.

Thanks to Jop and Boudewijn for hearing me out on my research and supporting me through the rough patches. Providing the much needed distractions at home.

I would like to thank Jessy, my sister, and my parents for keeping me motivated and having patience with the results.

Lastly, I would like to thank Karel with whom I have spend many hours on discord and teams working on our S&C projects during the lockdowns and finally on campus when corona was settling down.

Delft, University of Technology — Seymour Lubbers
August 6, 2023

Chapter 1

Introduction

The growing population and its accompanying increase in food demand [1], in combination with, global warming and decreasing arable land, results in a rapidly increasing greenhouse industry [2]. Greenhouses shelter crops, enable control over crop production, allow to boost production, permit the control of quality, allow for production of crops that would otherwise be impossible, and prolonging of the cultivation period [3]. Energy is needed for the climate control of the greenhouse, and water and nutrients, including CO_2 must be supplied from the outside.

Societal demands for fresh, safe, and price-worthy foods [4], in combination with sustainability concerns make it necessary to optimize greenhouse operations. A consequence of this combination of demands is that one can't simply increase inputs to grow better produce, this would dramatically lower the number of crops grown per unit of energy or nutrient input. One must exploit the outdoor environment, especially the sun, as much as possible. At the same time, it is not enough to only control the climate, the crops must also be taken into account [3].

Energy usually accounts for the second highest overhead cost in greenhouse crop production [5]. In the Netherlands, the cultivation of fruits and vegetables in greenhouses requires an enormous amount of energy. 77% of the total 100.5 PJ was due to heating and 23% due to electricity for supplemental lighting [6]. The Dutch horticultural industry has agreed with the Dutch government to decrease the total energy consumption [6].

A large discrepancy exists between the technologies used in commercial greenhouses and the state of research [3]. Controllers in commercial greenhouses tend to replicate the rules used by expert growers, while much of the research today is directed at the opportunities posed by optimal control. Optimal control, and its derivative MPC, have grabbed the attention of researchers for many years already [7]. Because this control method allows capturing system knowledge and, therefore, offers an opportunity to capture plant dynamics to a higher degree than growers themselves can [4].

A second opportunity for optimal control is offered by reinforcement learning. This machine learning approach can learn system dynamics by trial and error, also capturing system

uncertainties in the process. Being able to deal with system uncertainties is a welcome characteristic of reinforcement learning that standard MPC approaches do not offer. However, reinforcement learning also comes with its shortcomings, for one learning from trial and error dismisses any prior knowledge about the system and its dynamics.

Both approaches offer advantages that neatly overcome the disadvantages of the other strategy. Research towards combining the strengths of MPC and RL is rising and some promising results are achieved [8] [9] [10]. Nonetheless, little research can be found on combining MPC and RL in the field of greenhouse climate control up to this date.

1-1 Problem statement

The objective of this study is to improve the performance of greenhouse climate control in uncertain conditions, by using MPC as a function approximator for RL. Since model parameters can differ greatly from greenhouse to greenhouse and might not be known with great certainty, MPC can struggle with providing consistently good performance. This thesis aims to overcome the problems that parametric uncertainty poses when MPC is used to control the greenhouse climate, while maintaining the ability to incorporate system knowledge in the controller. This is done by using MPC as a function approximator in a Q-learning approach. Therefore, the main research question of this thesis is:

Can ENMPC using reinforcement learning be used in greenhouse climate control and improve performance compared to robust sample-based MPC in the presence of parametric uncertainty?

The performance will be assessed by computer simulations in a case study. In this case study the performance will be assessed in accordance with the performance indicators that are described in section 7-3. The effectiveness of multiple cost and reward functions will be compared. These variations will be applied in a framework known for its similarity with the real world.

To justify the main research question, the problem will be divided in multiple sub-questions:

- *Can ENMPC using RL decrease constraint violations, starting from the nominal model parameter values?*
- *Can ENMPC using RL increase economic performance, starting from perturbed initial model parameter values?*
- *Can ENMPC using RL increase economic performance and decrease constraint violations at the same time, while starting from perturbed initial model parameter values.*

These sub-questions address different objectives to which the controller can adhere. An overall well performing controller will also perform well on these sub-questions.

1-2 Thesis contribution

Enhancing greenhouse production processes to be more resource efficient, while at the same time increasing production can help decrease food shortages. Due to the adaptive properties of RL and the performance guarantees of MPC, combining the two in a data-driven ENMPC using RL setting has presented itself as a suitable alternative for greenhouse climate control. First, training examples for RL are scarce, since entire crop growth cycles are needed per parameter update. Second, MPC struggles to deal with the model uncertainty present in real greenhouse environments. This thesis aims to overcome these problems by combining MPC and RL to aid in their respective shortcomings, increasing economic performance and decreasing plant illness due to constraint violations.

With a growing population, decreasing resources and decreasing arable land, it will be necessary to increase food production while decreasing its cost to the environment and society. Therefore, this thesis is relevant for applying new theory to improve food production.

1-3 Thesis outline

The remainder of this thesis is structured as follows: chapter 2 discusses the necessary background on the modeling of greenhouses and the crops therein. In this chapter a greenhouse-crop model will be introduced. In chapter 3, the greenhouse-crop model will be elaborated on and the model used in this study is presented in detail. The uncertainty modeling used in this thesis will be discussed and the optimization problem solved by the MPC controller is presented. In chapter 4 and chapter 5 the reader is introduced to the concepts of MPC and RL. The application of function approximation will be introduced. chapter 6 will further delve into the specific method of function approximation that is relevant for this thesis. Here the theory supporting using ENMPC as a function approximator will be presented. In chapter 7 the first practical aspects of this thesis work are discussed. Most importantly, the model parameters are selected and the performance indicators for the system will be presented. chapter 8 will present the implementation of the control method on the system and discuss several case studies. Finally, the conclusions of this thesis and future research directions are presented and discussed in chapter 9. In Appendix C, a draft paper describing the findings of this thesis is included.

Chapter 2

Modeling and control of greenhouses and crops

As in many other fields, in greenhouse climate control, decision-making processes should be based on a clear and true picture of reality. Greenhouse modeling aims at mathematically describing horticultural greenhouses and the relationships between the outdoor and indoor climate, the effect of climate control equipment on the indoor climate, and the pivotal role of the cultivated crop [11]. This chapter starts with some general background information on greenhouses. Lastly, an overview of process-based greenhouse modeling is given and the van Henten model, which will be used in this study, is discussed in depth.

2-1 General background on greenhouses

Commercial greenhouses are complex systems with many different purposes, designs, and options for climate control equipment. A typical modern commercial greenhouse has a structure of steel or aluminum and is covered by glass, insulating sheets of multi-layer polycarbonate or foil [12]. [13] makes a distinction between four main types of greenhouses, namely: PE greenhouses, Parrel greenhouses, Chinese Solar Greenhouses, and Venlo type greenhouses. Examples of these greenhouses can be seen in 2-1a to 2-1d. The rest of this thesis work will focus on the Venlo-style greenhouse, as this is the most common commercially used greenhouse type in The Netherlands. The Venlo-type greenhouse is a tall, completely glass, greenhouse originating from The Netherlands. Venlo greenhouses are generally the most advanced and allow for various types of equipment to be used [13] [11].

2-2 Greenhouse modeling

The horticultural industry currently aims to improve yield and efficiency [14]. Efficiency in this case means the effective usage of resources, more yield is achieved with similar resource

(a) Venlo Greenhouse

(b) Parrel Greenhouse

(c) Gutter Connected Greenhouse

(d) Chinese Solar Greenhouse

Figure 2-1: Various styles of greenhouses in different parts of the world.

input. Models of greenhouses are set to play a pivotal role in addressing these problems. Increasingly, horticultural companies make use of mathematical models [15] [16] [17] [18]. Recently, artificial intelligence has been used to design control methods and create models [19] [20].

Generation of models

Generally, all models can be categorized into two categories. Models are either descriptive or process-based. [21] argues that the distinction between the two types of models is not always clear. Process-based models try to give a better understanding of the underlying principles in the system being modeled. Equations based on mathematical or statistical grounds are used by descriptive models to describe the underlying system, regardless of underlying principles [11]. Process-based models are built from the ground up. Process-based models start with the laws of physics and thermodynamics and often consist of two levels. The first level is the observed phenomena and the second, higher, level describes properties rising due to these phenomena [22]. Due to their origin in the physical principles behind the system, it is expected that process-based models will perform better outside the data range on which they have been validated than descriptive models would. [23] and [21] argue that process-based models can help bring insights that are relevant outside the range of the system they were designed on, predicting what would happen in unseen and possible scenarios.

In this work, a focus will be placed on process-based models. Descriptive models do not

provide many new insights into the system and the dynamics of the system are much harder to understand than when using process-based models. Within the class of process-based models, there is a wide array of models available in different model classes and complexity levels.

Process-based greenhouse models

[11] recognizes three classes of process-based greenhouse models in. The first class threats the greenhouse air as a perfectly stirred tank [24]. In this case, the air within the greenhouse is treated as an entity where spatial variability is ignored. All air within the greenhouse is uniform and is represented by a single value. This modeling approach often assumes the greenhouse to be infinitely large [25], causing the assumption that the indoor climate is not influenced by the outdoor air. The second class of models is built using energy simulation programs [26]. The programs are often originally built to simulate the energy demand of buildings and need many modifications to accurately apply them to greenhouses [27]. Lastly, [11] identifies a class of models based on Computational Fluid Dynamics (CFD) models. This class of models uses CFD to describe air movement within the greenhouse [28]. CFD analysis is computationally quite complex and significantly more computationally intensive than the perfectly stirred tank approach. The computational requirements of CFD limit their applicability [11]. CFD allows for the studying of airflow and temperature distributions and is usually done using either Ansys Fluent or COMSOL [26].

Even within these classes of greenhouse climate models a wide variety of complexity levels exist. For process-based models, the range of complexity levels is accurately shown by the models by van Henten and de Zwart [29] [30]. The complexity of the models is largely dependent upon the number of processes and objects that are included in them. More complex models include more processes and objects, and mechanistic descriptions of the processes are used, involving multiple influencing variables or inputs. Simpler models neglect some processes and/or objects and use more descriptive functions to summarize phenomena [11]. The van Henten model only includes two objects and four processes. Indoor and outdoor air are the objects, the processes involved are solar radiation, heating pipes, convection, and ventilation. The de Zwart model has 17 objects and multiple processes that were not involved in the van Henten model.

Next to the number of objects and processes involved, process-based models also vary in the way these objects and processes are described. The most commonly included processes are: Solar radiation and energy from lamps and other equipment, ventilation, convection and conduction, thermal radiation, energy losses to latent heat, transpiration, condensation, and crop photosynthesis [11].

In general, greenhouse climate is modeled separately from the crop growth and then integrated together. More details on crop growth modeling are provided in section 2-3.

The model objective provides some explanation for the vast range of models. Katzin et al. write that the more complex models are all devoted to the exploratory modeling of the system, while studies focusing on calibration and control tend to use simpler models. Especially, few objects for convective and Far Infrared (FIR) exchanges were used [11]. Van Henten and Bontsema also argue that it is still computationally favorable to keep the control model simple [32].

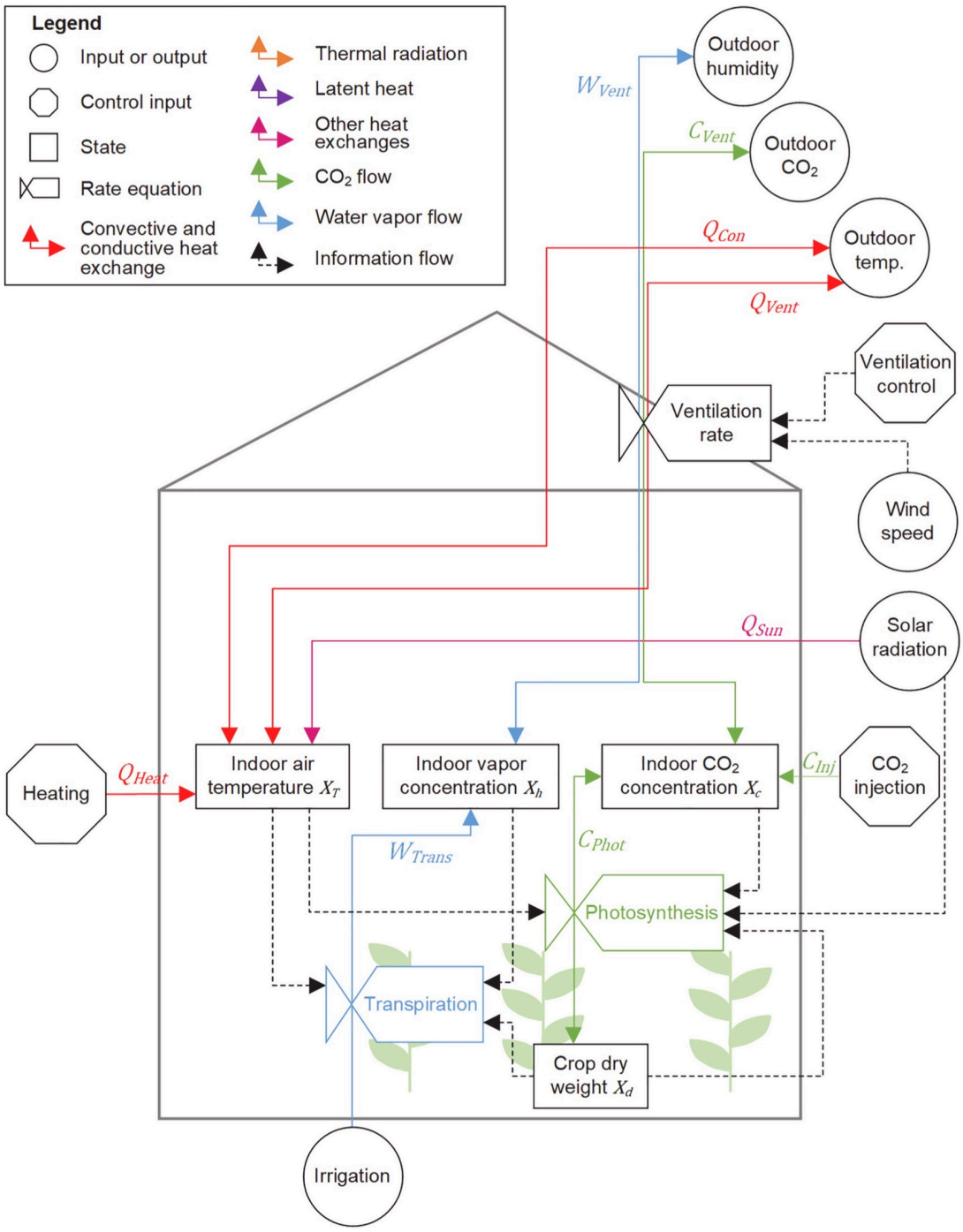

Figure 2-2: Figure taken from [11]. A scheme of the van Henten model [29] [31]. Solid lines indicate the mass and energy flows, and information is indicated by the dashed lines (see legend). Information flows indicate the influence of a control input or state on a flow rate. Valves over solid lines indicate rate equations that influence the flow rate passing through them. E.g., photosynthesis is influenced by solar radiation, crop dry weight, indoor air temperature, and indoor CO_2 concentration. Photosynthesis, in turn, influences the flow rate of CO_2 from the indoor air to the crop. Arrows indicate which objects act as sources, sinks, or both. E.g., the outdoor air may be both a sink or a source for CO_2, but CO_2 injection can only be a source.

Indoor climate description

The indoor climate can be described by its temperature, humidity, light, and CO_2 concentration. Some models only contain one attribute, and some contain multiple, the model used in this work contains the indoor temperature, humidity and CO_2 concentration. Generally, balances are considered to model these attributes. Incoming and outgoing balances are identified, for each of these balances. The net change for each of the attributes is then found by the difference between the incoming and outgoing flows [11]. The equations that describe these changes are Equation 2-1, Equation 2-2 and Equation 2-3.

$$\frac{dE}{dt} = (Q_{\text{Sun}} + Q_{\text{Heat}} + (Q_{\text{Lamp}} - Q_{\text{Vent}} - Q_{\text{Con}} - Q_{\text{Latent}} - Q_{\text{FIR}} - Q_{\text{Cool}}) \tag{2-1}$$

$$\frac{dM_{\text{W}}}{dt} = (W_{\text{Trans}} + W_{\text{Evap}} + W_{\text{Hum}} - W_{\text{Cond}} - W_{\text{Vent}} - W_{\text{Dehum}}) \tag{2-2}$$

$$\frac{dM_{\text{C}}}{dt} = (C_{\text{Inj}} - C_{\text{Phot}} - C_{\text{Vent}}) \tag{2-3}$$

In Equation 2-1, $\frac{dE}{dt}$ is the net change in energy in the greenhouse, with E representing the energy, Jm^{-2}, and t a unit of time. The right-hand side of the equation represents the incoming and outgoing flows, which may be positive (heating) or negative (cooling). In this case, the sign indicates the typical direction of the flow. Heating by the sun is represented by Q_{Sun}, heating by heating equipment is Q_{Heat} and by lamps Q_{Lamp}. Energy is generally lost due to ventilation, Q_{Vent}, convective and conductive exchanges with the outside climate, Q_{Con}, conversion from sensible to latent heat, Q_{Latent}, thermal radiation, Q_{FIR} and heat extracted by cooling equipment, Q_{Cool}.
In Equation 2-2, $\frac{dM_{\text{W}}}{dt}$ is the net change in water vapor mass in the greenhouse. Incoming flows are typically from crop transpiration, W_{Trans}, water evaporation from surfaces such as the soil, W_{Evap}, humidification by control equipment, W_{Hum}. Outgoing flows typically consist of Condensation on cold surfaces, W_{Cond}, water vapor lost due to ventilation, W_{Vent}, humidification by control equipment, W_{Dehum}.
In Equation 2-3, $\frac{dM_{\text{C}}}{dt}$ is the net change in CO_2 concentration in the greenhouse. The only incoming flow results from CO_2 injected by control equipment, C_{Inj}. Outgoing flows typically consist of CO_2 used due to photosynthesis by the crops, C_{Phot} and losses due to ventilation C_{Vent}.

Van Henten model

In this study the van Henten model is used, an example of a process-based model, based on differential equations describing some of the energy balances in the system [29] [31]. The model is composed of 4 states, all described by separate differential equations. Three of these states correspond to energy balances of the indoor temperature, X_{T} (°C), indoor vapor concentration, X_{h} (kg {water vapor} m^{-2}), and the indoor CO_2 concentration, X_{c} (kg {CO_2} m^{-2}). The fourth state is concerned with the crop, this state is the crop dry-weight, X_{d} (kg {dry weight} m^{-2}). A summary of the differential equations of the van Henten model is shown in Equation 2-4. The first three lines in Equation 2-4 represent the balance of energy, water vapor and CO_2, respectively. $\frac{dX_{\text{T}}}{dt}$ gives the energy balance, resulting in a

temperature change over time. Energy enters the system due to heat coming from the sun and the heaters, it leaves the system due to ventilation and convection. $\frac{dX_\mathrm{h}}{dt}$ is the water vapor balance and consists of water vapor entering the system due to crop transpiration and leaving due to ventilation. $\frac{dX_\mathrm{c}}{dt}$ is the CO_2 balance, consisting of CO_2 being injected into the system and leaving the system due to ventilation and crop photosynthesis. The functions $f(X_\mathrm{T}, X_\mathrm{c}, X_\mathrm{d}, V_\mathrm{rad})$, $g(X_\mathrm{T}, X_\mathrm{c}, X_\mathrm{d}, V_\mathrm{rad})$ and $h(X_\mathrm{T}, X_\mathrm{h}, X_\mathrm{d})$ are described in [29] and [31], the parameters $c_{\mathrm{cap,q}}$, $c_{\mathrm{cap,q}}$ and $c_{\mathrm{cap,q}}$ are also described there. The exact implementation used in this research is discussed in chapter 3.

$$\begin{aligned}
\frac{dX_\mathrm{T}}{dt} &= \frac{1}{c_{\mathrm{cap,q}}}(Q_\mathrm{sun} + Q_\mathrm{heat} - Q_\mathrm{vent} - Q_\mathrm{con}), && (^\circ\mathrm{C}s^{-1}) \\
\frac{dX_\mathrm{h}}{dt} &= \frac{1}{c_{\mathrm{cap,h}}}(W_\mathrm{trans} - W_\mathrm{vent}), && (kgm^{-2}s^{-1}) \\
\frac{dX_\mathrm{c}}{dt} &= \frac{1}{c_{\mathrm{cap,c}}}(C_\mathrm{inj} - C_\mathrm{vent} - C_\mathrm{phot}), && (kgm^{-2}s^{-1}) \\
\frac{dX_\mathrm{d}}{dt} &= f(X_\mathrm{T}, X_\mathrm{c}, X_\mathrm{d}, V_\mathrm{rad}), && (kgm^{-2}s^{-1}) \\
C_\mathrm{Phot} &= g(X_\mathrm{T}, X_\mathrm{c}, X_\mathrm{d}, V_\mathrm{rad}), && (kgm^{-2}s^{-1}) \\
W_\mathrm{Trans} &= h(X_\mathrm{T}, X_\mathrm{h}, X_\mathrm{d}), && (kgm^{-2}s^{-1})
\end{aligned} \tag{2-4}$$

2-3 Crop growth modeling

In the previous section an emphasis was placed on greenhouse climate modeling. The Van Henten model was provided, in which a crop growth model is integrated. The modeling of the crop is done through X_d. In this section the crop growth modeling will be further delved into.

Crop growth modeling is an important aspect of modeling a greenhouse production system. The crop is the final product, and the entity the greenhouse is controlled for. Crop modeling tries to model crop growth as a function of irradiation, temperature and crop characteristics [33]. In this work a focus will be placed on the modeling and simulation of lettuce growth. Many different lettuce growth simulation modeling efforts have been undertaken over the years. The first structural and storage pool was defined in [34], following the approach by de Wit, is was later also described in [35]. Two state variables were considered, the structural and non-structural dry matter. Prior to this, dry mass production was considered as the difference between photosynthesis and respiration in [36]. In [37], a simple model for lettuce growth in an environment with elevated carbon dioxide concentrations was described. In [38], the work from [37] and [34] was combined. Later, [34] was adapted for greenhouse conditions in [39] and [29]. In [33], the Simple and Universal Crop Growth Simulator (SUCROS87) was developed. The model formulation of SUCROS87 was also used in [29]. Both [40] and [41] used SUCROS87 to develop greenhouse models. Based on the work in [29], [34], and [42], the Nitrate Control in Lettuce (NiCoLet) model was developed in [43] and improved upon in [44], [45], [46], and [47], respectively. In this work the lettuce growth model that is integrated in the Van Henten model will be further discussed.

The Van Henten model describes crop growth by a single state variable, namely the crop dry

weight [29]. The model follows the SUCROS-87 formulation of crop growth, in which the basis for calculating dry matter production is the canopy net carbon dioxide assimilation rate [33]. Dry matter production in the crop can then be described by:

$$\frac{dX_{\mathrm{d}}}{dt} \;=\; c_{\beta}\big(c_{\alpha}\phi_{\mathrm{phot}} - \phi_{\mathrm{resp}}\big), \qquad \text{kg m}^{-2}\ \text{s}^{-1} \tag{2-5}$$

in which ϕ_{phot} is the gross carbon dioxide uptake due to canopy photosynthesis, and ϕ_{resp} is the maintenance respiration rate, both in terms of the amount of carbohydrates consumed. c_{α} and c_{β} are parameters that convert assimilated carbon dioxide into sugar equivalents and the yield factor that accounts for the respiratory and synthesis losses during the conversion of carbohydrates to structural material, respectively. c_{β} has a value between zero and one.

The gross canopy photosynthesis is described by:

$$\phi_{\mathrm{phot}} \;=\; \phi_{\mathrm{phot,max}}\Big(1 - e^{-c_{\mathrm{k}}c_{\mathrm{lar,d}}(1-c_{\tau})X_{\mathrm{d}}}\Big), \qquad \text{kg m}^{-2}\ \text{s}^{-1} \tag{2-6}$$

an empirical relation in which $\phi_{phot,max}$ is the gross carbon dioxide assimilation rate of the canopy having an effective canopy surface of 1 m^2 per square meter of soil at complete soil covering [29]. $1 - e^{-c_{\mathrm{k}}c_{lar,d}(1-c_{\tau})X_{\mathrm{d}}}$ accounts for the geometrical and optical properties of the canopy. All environmental effects are captured in the value of $\phi_{phot,max}$, which is described by:

$$\phi_{\mathrm{phot,max}} = \frac{\epsilon c_{\mathrm{par}} c_{\mathrm{rad,rf}} V_1 \sigma_{\mathrm{CO_2}} (X_{\mathrm{C}} - \Gamma)}{\epsilon c_{\mathrm{par}} c_{\mathrm{par,rf}} V_1 + \sigma_{\mathrm{CO_2}} (X_{\mathrm{C}} - \Gamma)} \tag{2-7}$$

in which ϵ, in Kg J^{-1}, is the light use efficiency. c_{par} is the ratio of photosynthetically active radiation to total solar radiation, $c_{\mathrm{rad,rf}}$ is the transmission ratio of the roof for solar radiation, V_1, in W m^{-2}, is the solar radiation outside the greenhouse, $\sigma_{\mathrm{CO_2}}$, in m s^{-1}, is the canopy conductance for carbon dioxide transport into the leaves, X_{C}, in kg m^{-3}, is the carbon dioxide concentration, and Γ, in kg m^{-3}, is the carbon dioxide compensation point which accounts for photorespiration at high light levels, which is also affected by the temperature, X_{t}, in °C. From this it can be seen that the temperature and carbon dioxide concentration, as well as the radiation, inside a greenhouse directly affect the growth of the crop.

The maintenance respiration rate of the crop is given by:

$$\phi_{resp} \;=\; \big(c_{\mathrm{resp,s}}(1 - c_{\tau}) + c_{\mathrm{resp,r}}c_{\tau}\big)\, X_{\mathrm{d}}\, c_{\mathrm{Q10,resp}}^{\ \big(X_{\mathrm{t}}-25\big)/10}, \qquad \text{kg m}^{-2}\ \text{s}^{-1} \tag{2-8}$$

where $c_{resp,s}$ and $c_{resp,r}$ are the maintenance respiration rates for the shoots and roots at 25°C, respectively, expressed in mass of glucose consumed. $c_{Q10,resp}$ is the Q_{10}-factor for the maintenance respiration [29].

Chapter 3

Greenhouse and lettuce model

In this chapter, the greenhouse and lettuce model will be presented. In section 3-1 the model will be presented, section 3-2 discusses the type of uncertainty that will be taken into account in this work. section 3-3 further discusses the state propagation and measurement functions of the model. Finally, section 3-4 discusses the optimization problem that needs to be solved.

3-1 The model

The model that is used in this work is the nonlinear model first presented in [29] and discretized using the fourth-order Runge Kutta method. This results in

$$x(k+1) = f(x(k),\ u(k),\ d(k), p), \tag{3-1}$$
$$y(k) = g(x(k),\ p) \tag{3-2}$$

with discrete time $k \in \mathbb{Z}^{0+}$, state $x(k) \in \mathbb{R}^4$, input $u(k) \in \mathbb{R}^3$, disturbance $d(k) \in \mathbb{R}^4$and $f(\cdot)$, $g(\cdot)$ nonlinear functions. Further details on the state propagation and measurement functions can be found in section 3-3. The state, input and disturbance signals are further defined as:

$$x(k) = \begin{pmatrix} W(k) & C_{\mathrm{CO_2,a}}(k) & T_{\mathrm{a}}(k) & C_{\mathrm{H_2O,a}}(k) \end{pmatrix}^T \tag{3-3}$$

$$= \begin{pmatrix} x_1(k) & x_2(k) & x_3(k) & x_4(k) \end{pmatrix}^T \tag{3-4}$$

$$u(k) = \begin{pmatrix} u_{\mathrm{CO_2,o}}(k) & u_{\mathrm{v}}(k) & u_{\mathrm{q}}(k) \end{pmatrix}^T \tag{3-5}$$

$$= \begin{pmatrix} u_1(k) & u_2(k) & u_3(k) \end{pmatrix}^T \tag{3-6}$$

$$d(k) = \begin{pmatrix} I(k) & C_{\mathrm{CO_2,o}}(k) & T_{\mathrm{o}} & C_{\mathrm{H_2O,o}}(k) \end{pmatrix}^T \tag{3-7}$$

$$= \begin{pmatrix} d_1(k) & d_2(k) & d_3(k) & d_4(k) \end{pmatrix}^T \tag{3-8}$$

The state $x(k)$ contains the dry matter content of the lettuce, W, in $\mathrm{kg} \cdot \mathrm{m}^{-2}$, which is the lettuce's weight per square meter, after all its water has been removed. The crop dry matter is often used because it shows less seasonal variation than fresh weight. Furthermore, the state contains the indoor carbon dioxide concentration, $C_{\mathrm{CO_2,a}}$, in $\mathrm{kg} \cdot \mathrm{m}^{-3}$, air temperature, T_{a}, in °C, and humidity, $C_{\mathrm{H_2O,a}}$, in $\mathrm{kg} \cdot \mathrm{m}^{-3}$.

The control input $u(k)$ contains the rate of carbon dioxide supply, $u_{\mathrm{CO_2}}$, in $\mathrm{mg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$, ventilation rate through the window vents, u_{v}, in $\mathrm{mm} \cdot \mathrm{s}^{-1}$, and energy supply by the heaters, u_{q}, in $\mathrm{W} \cdot \mathrm{m}^{-2}$.

The weather disturbance, $d(k)$, contains the radiation from the sun, I_{o}, in $\mathrm{W} \cdot \mathrm{m}^{-2}$, the outside carbon dioxide concentration, $C_{\mathrm{CO_2,o}}$, in $\mathrm{kg} \cdot \mathrm{m}^{-3}$, outdoor temperature T_{o}, in °C, and humidity, $C_{\mathrm{H_2O,o}}$, in $\mathrm{kg} \cdot \mathrm{m}^{-3}$.

The measurements, $y(k) \in \mathbb{R}^4$, contain the dry weight matter, W, in $\mathrm{g} \cdot \mathrm{m}^{-2}$, the indoor carbon dioxide concentration, $C_{\mathrm{CO_2,a}}$, in ppm, the indoor air temperature, T_{a}, in °C, and relative humidity, $C_{\mathrm{H_2O,a}}$, in %. It should be clear that $y(k)$ contains equivalent states as $x(k)$, but with different units for the dry weight, indoor carbon dioxide concentration and humidity.

In Figure 3-1, the greenhouse model with lettuce is graphically depicted. The interaction between the states, control inputs, and disturbances is indicated by arrows. From this figure it is clear that the control inputs and weather disturbances influence the greenhouse climate, which in turn influences the lettuce. The only exception to this is the solar radiation, which also has a direct influence on the crop. All three control inputs have a direct influence on the greenhouse climate, while having an indirect influence on the lettuce's dry weight. It is via controlling the greenhouse climate that the lettuce growth is controlled.

3-2 Uncertainty modeling

In the model, various sources of uncertainty can be introduced. In this section some sources of uncertainty will be discussed, and, finally, a conclusion will drawn which details the kind of uncertainty taken into account in this work.

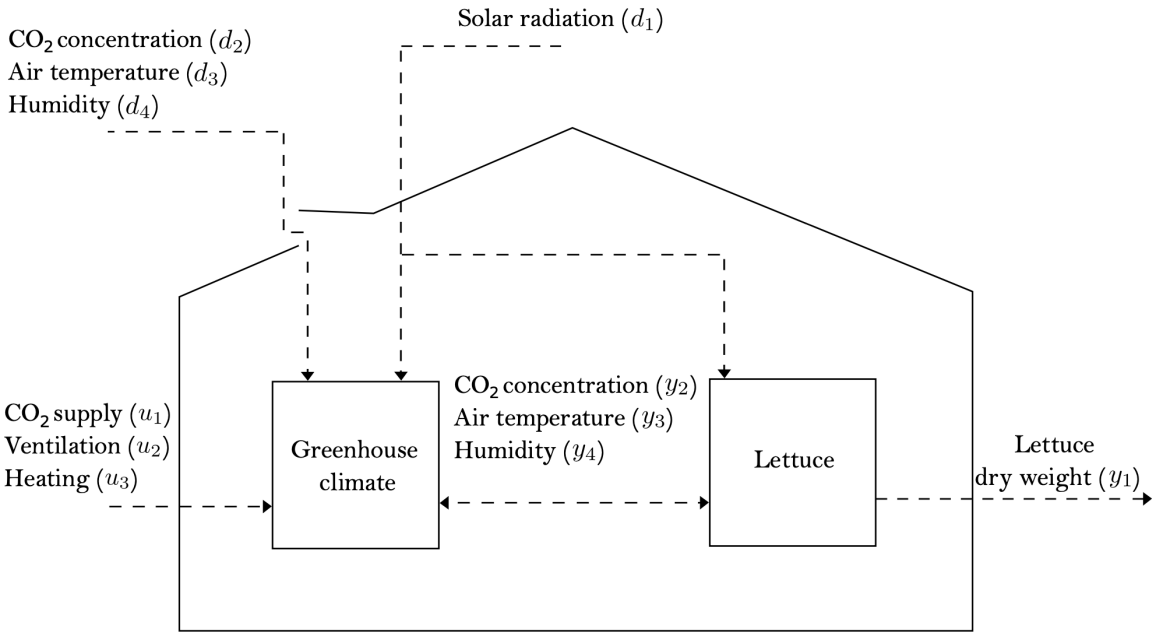

Figure 3-1: Greenhouse with lettuce schematically represented. The interaction between the greenhouse and lettuce model is indicated with the arrows. The influence of the control signal $u(k)$ and weather disturbance $d(k)$ are also indicated by the arrows. The figure is taken from [48].

In a real world application uncertainty exists in all aspects of the model. The weather disturbances are not exactly known, the control inputs are not perfectly exact, the greenhouse and lettuce model is not deterministic, the state measurements will not be perfect, and further aspects influencing the cost function might also not be known exactly.

Model mismatch between the actual system and the model may cause considerable prediction errors [49]. This mismatch is a results from the simplification that has been done to make the process less computationally intensive, or simply because the the system is not perfectly known. Discretizing a continuous system can introduce a discretization error. All factors combined result in parametric uncertainty, which has been discussed in [49] and will be investigated in this work.

Uncertainty in this work

One of the goals of this work is to compare the performance of a data-driven economic NMPC using RL with the performance achieved by robust sample-based nonlinear MPC. In order to be able to compare the different controllers it is important to keep all conditions that apply to the system the same. The kind of uncertainty that will be considered in this work is parametric uncertainty, equal to the uncertainty considered in [49]. In this work, it is assumed that the model parameters can be modeled as an uncertain parameter with known probabilistic properties. It is assumed that:

$$\hat{p} \sim \mathcal{U}\Big(\mu_{\hat{\mathrm{p}}}, \Sigma_{\hat{\mathrm{p}}}(k)\Big), \tag{3-9}$$

with uniform distribution $\mathcal{U}(\cdot)$ having a mean value of $\mu_p = p$ and co-variance matrix $\Sigma_{\hat{p}} = \mathbb{E}[(\hat{p} - \mu_{\hat{p}})(\hat{p} - \mu_{\hat{p}})^T]$. with expectation operator $\mathbb{E}$. Furthermore, it is assumed that $\Sigma_{\hat{p}}$ is a diagonal matrix, meaning that there is no cross-covariance. The justification for a uniform distribution is two-folded. Firstly, certain parameters can not be negative or outside certain ranges. Secondly, it is undesired to have an infinite support (such as a Gaussian distribution). Due to the assumption of having an uncertain parameter in the measurement model, the state and measurement estimation are also uncertain. The system model is then depicted as:

$$x(k+1) = f(\hat{x}(k),\ u(k),\ d(k), \hat{p}), \tag{3-10}$$
$$y(k) = g(\hat{x}(k),\ \hat{p}) \tag{3-11}$$

where $\hat{p}$ indicates the uncertain parameters.

In the case studies, in some cases, it can become useful to force a large parameter offset from the nominal parameter values. This can be forced with an equal flat offset on all model parameter values, e.g. all parameters can be increased by 10%. Although this deviation in parameter value is not as true to reality, it is helpful in creating meaningful case studies.

3-3 State propagation and measurement functions

In this section, the nonlinear functions $g(\cdot)$ and $f(\cdot)$, from Equation 3-1, are presented. The function $f(\cdot)$, represents the state propagation model, and is defined as follows:

$$x_1(k+1) = x_1(k) + h\Big(c_{1,1}\phi_{\text{phot,c}}(k) - c_{1,2}x_1(k)2^{x_3(k)/10-5/2}\Big) \tag{3-12}$$
$$x_2(k+1) = x_2(k) + \frac{h}{c_{2,1}}\Big(-\phi_{\text{phot,c}}(k) + c_{2,2}x_1(k)2^{x_3(k)/10-5/2} + u_1(k)10^{-6} - \phi_{\text{vent,c}}(k)\Big) \tag{3-13}$$
$$x_3(k+1) = x_3(k) + \frac{h}{c_{3,1}}\Big(u_3(k) - (c_{3,2}u_2(k)10^{-3} + c_{3,3})(x_3(k) - d_3(k)) + c_{3,4}d_1(k)\Big) \tag{3-14}$$
$$x_4(k+1) = x_4(k) + \frac{h}{c_{4,1}}\Big(\phi_{\text{transp,h}}(k) - \phi_{\text{vent,h}}(k)\Big) \tag{3-15}$$

with

$$\phi_{\text{phot,c}}(k) = \Big(1 - exp(-c_{1,3}x_1(k))\Big)\Big(c_{1,4}d_1(k)(-c_{1,5}x_3(k)^2 + c_{1,6}x_3(k) - c_{1,7})(x_2(k) - c_{1,8})\Big)/\varphi(k), \tag{3-16}$$
$$\varphi(k) = c_{1,4}d_1(k) + (-c_{1,5}x_3(k)^2 + c_{1,6}x_3(k) - c_{1,7})(x_2(k) - c_{1,8}), \tag{3-17}$$
$$\phi_{\text{vent,c}}(k) = (u_2(k)10^{-3} + c_{2,3})(x_2(k) - d_2(k)), \tag{3-18}$$
$$\phi_{\text{vent,h}}(k) = (u_2(k)10^{-3} + c_{2,3})(x_4(k) - d_4(k)), \tag{3-19}$$
$$\phi_{\text{transp,h}}(k) = c_{4,2}\Big(1 - exp(-c_{1,3}x_1(k))\Big)\Big(\frac{c_{4,3}}{c_{4,4}(x_3(k) + c_{4,5}}exp\Big(\frac{c_{4,6}x_3(k)}{x_3(k) + c_{4,7}}\Big) - x_4(k)\Big) \tag{3-20}$$

parameter	value	parameter	value	parameter	value	parameter	value
$c_{1,1}$	0.544	$c_{2,1}$	4.1	$c_{3,1}$	$3 \cdot 10^4$	$c_{4,1}$	4.1
$c_{1,2}$	$2.65 \cdot 10^{-7}$	$c_{2,2}$	$4.87 \cdot 10^{-7}$	$c_{3,2}$	1290	$c_{4,2}$	0.0036
$c_{1,3}$	53	$c_{2,3}$	$7.5 \cdot 10^{-6}$	$c_{3,3}$	6.1	$c_{4,3}$	9348
$c_{1,4}$	$3.55 \cdot 10^{-9}$	$c_{2,4}$	75560	$c_{3,4}$	0.2	$c_{4,4}$	8314
$c_{1,5}$	$5.11 \cdot 10^{-6}$	$c_{2,5}$	273.15			$c_{4,5}$	273.15
$c_{1,6}$	$2.3 \cdot 10^{-4}$	$c_{2,6}$	101325			$c_{4,6}$	17.4
$c_{1,7}$	$6.29 \cdot 10^{-4}$	$c_{2,7}$	0.044			$c_{4,7}$	239
$c_{1,8}$	$5.2 \cdot 10^{-5}$					$c_{4,8}$	17.269
						$c_{4,9}$	238.3

Table 3-1: Nominal values of the model parameters taken from [29].

with h the sample time. $\phi_{\text{phot,c}}(k)$, $\phi_{\text{vent,c}}(k)$, $\phi_{\text{transp,h}}(k)$, and $\phi_{\text{vent,h}}(k)$ are the gross canopy photosynthesis rate, mass exchange of CO_2 through the vents, canopy transpiration and mass exchange of H_2O through the vents, respectively.

The initial state and control signals are $x(0) = (0.0035\ 0.001\ 15\ 0.008)^T$, $u(0) = (0\ 0\ 0)^T$.

The measurement equation, $g(\cdot)$, is defined as:

$$y_1(k) = 10^3 x_1(k), \qquad g \cdot m^{-2}, \tag{3-21}$$

$$y_2(k) = \frac{c_{2,4}\big(x_3(k)+c_{2,5}\big)}{c_{2,6}c_{2,7}} \cdot x_2(k), \qquad ppm \cdot 10^3, \tag{3-22}$$

$$y_3(k) = x_3(k), \qquad ^{\circ}\text{C}, \tag{3-23}$$

$$y_4(k) = \frac{c_{2,4}\big(x_3(k)+c_{2,5}\big)}{exp\Big(\frac{c_{4,8}x_3(k)}{x_3(k)+c_{4,9}}\Big)} \cdot x_4(k), \qquad \%, \tag{3-24}$$

The model parameters are chosen as in [29], and given in Table 3-1.

3-4 Optimization problem

The goal is to achieve maximum dry matter production, with the least amount of energy input as possible. It is assumed that at each time instant, the state $x(k)$ can be measured. Then, the following optimization problem is solved at each time step:

$$\begin{aligned}
&\min_{u(k)} \sum_{i=1}^{Ns} \sum_{k=k_0}^{k_0+N_P} V\big(u(k), \hat{y}^i(k)\big), \\
&\text{s.t. } \textit{Equation } 3-1,\ \hat{x}(k_0) = \hat{x}_0, \\
&\quad u_{\min} \le u(k) \le u_{\max}, \\
&\quad |u(k) - u(k-1)| \le \delta u, \\
&\quad \hat{y}_{\min}(k) \le \hat{y}^i(k) \le \hat{y}_{\max}(k), \text{for } i = 1, \ldots, N_s, \\
&\quad \text{and for } k = k_0, \ldots, k_0 + N_P,
\end{aligned} \tag{3-25}$$

with $N_P \in \mathbb{R}$ the prediction horizon. The value $N_s \in \mathbb{R}$ represents the number of realisations taken from Equation 3-9. In this work the number of realisations taken from the distribution will be 1. The comparison work utilizes sample-based robust control and takes multiple realisations.

The constraints in Equation 3-25 are defined:

$$u_{\min} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T, \tag{3-26}$$

$$u_{\max} = \begin{pmatrix} 1.2 & 7.5 & 150 \end{pmatrix}^T, \tag{3-27}$$

$$\delta u = \frac{1}{10} u_{\max}, \tag{3-28}$$

$$\hat{y}_{\min} = \begin{pmatrix} 0 & 0 & f_{\hat{y}_{3,\min}}(k) & 0 \end{pmatrix}^T, \tag{3-29}$$

$$\hat{y}_{\max} = \begin{pmatrix} \inf & 1.6 & f_{\hat{y}_{3,\max}}(k) & 70 \end{pmatrix}^T, \tag{3-30}$$

with lower and upper bound on the control signal defined by $u_{\min}$ and $u_{\max} \in \mathbb{R}^3$, and the bound on the change of the control signal defined by $\delta u \in \mathbb{R}^3$. These bounds correspond to current horticultural practice. The lower and upper bound on the output are $\hat{y}_{3,\min}(k)$ and $\hat{y}_{3,\min}(k) \in \mathbb{R}^4$. In these bounds, only the third element is time-varying and defined as:

$$\begin{aligned} f_{\hat{y}_{3,\min}}(k) &= \begin{cases} 15, & \text{if } d_1(k_0) < 10 \\ 20, & \text{otherwise.} \end{cases}, \\ f_{\hat{y}_{3,\max}}(k) &= \begin{cases} 20, & \text{if } d_1(k_0) \geq 10 \\ 25, & \text{otherwise.} \end{cases} \end{aligned} \tag{3-31}$$

The time varying constraints are on the indoor temperature are set such that it is colder during the night than during the day. A consequence is that during the night, when the outside temperature is colder, the controller is not working as hard to keep the temperature at a high level. Since, if there is any, the sun is heating up the greenhouse during the day, less additional heating is necessary during this period to achieve relatively high temperatures in the greenhouse hence energy can be saved [48].

Chapter 4

Model predictive control

The current state of greenhouse climate control in industry is relies mostly on the expert knowledge of the growers themselves, these systems do not contain much knowledge about the system or the growth of the crops. Most systems in the field today still try to mimic the knowledge of the growers and consist mostly of a set of rules on what to do in what situation. The state of academic research on the topic of greenhouse climate control is much more advanced, it makes more use of models and knowledge of the system itself.

In this chapter, a short general introduction to MPC will be given. Then, the consequences of parametric uncertainty are delved into.

4-1 Introduction to MPC

For over 30 years already, research has seen an opportunity in optimal control of greenhouse climates [7]. Optimal control tries to satisfy as many goals, set by the grower, as possible. The goals that are set by the grower are usually the set points for the greenhouse climate in this case, e.g. temperature, humidity, etc. Given a system with an initial state $x(t_0) = x_0$ and auxiliary output variables

$$z(t) = h(x(t), u(t), d(t)) \tag{4-1}$$

in which $z(t)$ represents the auxiliary variables, $x(t)$ denotes the states, $u(t)$ denotes the inputs, $d(t)$ denotes the disturbances, and $h(t)$ denotes a potentially nonlinear function. Then find the control input

$$u(t), t_0 \leq t \leq t_f \tag{4-2}$$

such that the objective function, J, is minimized. In the objective function, defined as below, Φ represents the terminal cost, and $L()$ represents the stage cost.

$$J(u(t), t_0, t_f) = \Phi(x(t_f), t_f) + \int_{\tau=t_0}^{\tau=t_f} L(z(\tau))d\tau \tag{4-3}$$

Finally, additional inequality constraints must be met:

$$c\Big(x(t), u(t), d(t)\Big) \leq 0, \quad t_0 \leq t \leq t_f \tag{4-4}$$

The constraint conditions, in the case of greenhouse climate control, may refer to input constraints, imposed by the operating range of the equipment, state constraints, such as the maximum allowable temperature range, and output constraints, such as the maximum allowable vapor concentration [3]. It should be noted however, that in the case of complete models, correctly capturing all crop phenomena, output, and state constraints would not be necessary. If for instance a too high crop temperature is detrimental to the crop, and this is accurately captured in the model, the controller would steer the system away from these temperatures. Currently, such knowledge is often still captured by the system constraints alone. On the other hand, input constraints that have to do with equipment limitations are always needed.

Optimal control is often implemented in the form of model predictive control (MPC). The basic concept of MPC is to use a dynamic model of a system in order to predict system behavior, and optimize the forecast to produce the best decision possible, e.g. to compute the best input [50]. A model of the system is used by MPC to predict the future output of the system, given a series of inputs. Feedback is provided, as the state prediction starts from the current state. After optimization, the first calculated input is applied, and the cycle repeats. MPC provides solutions that are optimal from the control point of view [3]. The MPC controller, also called a receding horizon controller, can work with multiple control objectives and keep constraints on input, state and output into account. The optimization problem that MPC solves in greenhouse climate control setting is:

$$\begin{aligned}
\min_{u} \quad & V(u_k, y_k), && \text{(4-5)}\\
\text{s.t.} \quad & x_{k+1} = f(\hat{x}_k, u_k, d_k, \hat{p}), && \text{(4-6)}\\
& y_k = g(\hat{x}_k, \hat{p}), && \text{(4-7)}\\
& y_{\text{k,min}} \leq y_k \leq y_{\text{k,max}}, && \text{(4-8)}\\
& |u_k - u_{k-1}| \leq \delta u, && \text{(4-9)}\\
& u_{\min} \leq u_k \leq u_{\max}, && \text{(4-10)}\\
& \hat{x}_0 = x_0, && \text{(4-11)}
\end{aligned}$$

The MPC controller aims to minimize the cost associated to the problem, given by Equation 4-5. The controller does this by changing the input, u_k. The problem is subject to model dynamics Equation 4-6 and Equation 4-7, dependent on the state, x_k, input, u_k, disturbances, d_k, and parameters, p. The parameters and state are uncertain, noted by the $\hat{}$ symbol. The problem is subject to constraints Equation 4-8, Equation 4-9, Equation 4-10 and Equation 4-11.

In the case of greenhouse climate control, a trade-off has to be made when selecting a model for predicting future states. There will be a trade-off between the accuracy and complexity of the model. As discussed earlier, in an ideal scenario, the model should capture all phenomena and optimization would then automatically steer the system to avoid harm to the crops.

However, the complexity of models that capture more phenomena makes them less suited for optimization. Using accurate, and thus complex, models requires large amounts of computing power and time. Because greenhouse process models can become quite elaborate [11], the optimization procedure can take longer than the fastest time scales within the model. For control purposes, less complex and also less accurate models have been developed that still capture the most important phenomena. An example is the model by van Henten [29], discussed in chapter 2 and used in this study.

MPC comes in many varieties, not just in the form mentioned above. The first distinction can be made between linear and non-linear MPC. MPC can be implemented in a way that assumes the system to be deterministic, while other methods include robust MPC or stochastic MPC, which incorporate uncertainty in the model. A detailed explanation of how these methods work falls outside the scope of this paper, interested readers are referred to [50].

Optimal control is achieved with the use of MPC if all system dynamics are deterministic. However, as discussed in chapter 3, at least some of the systems in a greenhouse are stochastic and will need some way of handling, e.g. parametric uncertainty. In the next section, the handling of uncertainty by MPC is discussed.

4-2 MPC in case of uncertainty

Taking the uncertainty in the system into account will always cause the system to be less efficient than if the system is deterministic. Boersma et al. explore the relation between parametric uncertainty and performance in [49]. It was found that a relative parameter uncertainty of 20% reduced crop yield by 11% and demands for CO_2, ventilation, and heating changed drastically.

Parametric uncertainty

[50] makes a distinction between three different types of uncertainty. Models can have an additive disturbance that is unknown, the state of the system might not be accurately known, or the system model for control is inaccurate [50]. In this work the system model for control is inaccurate, the parameters of the model are not known precisely.

A system that has parametric uncertainty can be modeled as

$$x^+ = f(x, u, \theta) \tag{4-12}$$

in which θ are the parameters of the system and are only known to belong to a compact set Θ. An example that is often studied is

$$x^+ = Ax + Bu \tag{4-13}$$

in which $\theta = (A, B)$ may take any value in Θ. In greenhouse climate control, parametric uncertainty is always present. Often, in literature, the state is assumed to be deterministic to lessen the issue of parametric uncertainty.

Results are obtained in [49], a non-linear sample-based robust MPC is proposed for a greenhouse with crops under parametric uncertainty. A comparison is made with a deterministic

case, and it is found that performance is increasingly lost with increasing uncertainty, where the performance is measured as the crop dry weight production with respect to the energy and CO_2 input. Next to the significant decrease in the performance, a notable change in system inputs is seen. Parametric uncertainty of 20% results in a performance loss of 11.2%. CO_2 injection is decreased by 79.7%, while ventilation and heating are increased by over 90%. All these results increase with increasing parametric uncertainty.

Chapter 5

Reinforcement learning

Reinforcement learning (RL) control methods are based on the interaction between an agent and its environment. Complex decision-making problems have been solved using RL methods, it is able to learn strategies that humans don't easily comprehend. The greenhouse climate control problem can be regarded as a decision-making problem, where at each time step an action has to be performed. This action, in the form of the inputs to the system, has to be decided upon such that the crop grows optimally with the least possible input cost. This chapter will go over the possibilities of RL in greenhouse climate control. First, the concept of RL will be briefly introduced. Then, function approximation will be introduced. The use of function approximation is central to the research in this study and will be further elaborated on in chapter 6.

5-1 Introduction to reinforcement learning

RL is a method of machine learning, which can interactively solve complex optimization and control problems. In RL there is an agent and an environment. The environment is the space in which the task is defined, and the agent is the decision maker [51]. The environment gives the agent a reward that is based on the quality of the action that is taken, and the agent tries to maximize this reward in time. By iterating many times over the possible states the agent is eventually able to find the optimal control strategy [51]. RL falls between supervised and unsupervised learning, as the correct action is not known, but a reward can be given based on the output of that action. RL is inspired by human and animal learning, the rewards are used as feedback on the taken actions.

In the RL framework, the environment is modeled as a Markov Decision Process (Markov Decision Process (MDP)). An MDP is a tuple of the states, inputs, the state transition function, and the reward function.

- X is the finite state space
- U is the finite action space

- $f : X \times U \longrightarrow X$ is the state transition function
- $\rho : X \times U \longrightarrow \mathbb{R}$ is the rewards function

The Markov property states that the future state only depends on the current state and the action taken, future states are not dependent on previous states. The future is independent of the past, given the present.

The agent, the controller, is a state feedback controller and learns the optimal mapping from states to actions, from this it learns an optimal control law (π). The learning goal of any RL controller is to find the policy that maximizes the return. The return is defined as a combination of the immediate and future rewards, and the agent can't know the future rewards yet, a value function, V, is introduced that defines the expected return of following a certain policy, π, for each state or state-action pair. The policy tells what action should be taken in what state. In state s, the state-value function for an agent following a deterministic policy π is defined for discounted rewards as:

$$V^{\pi}(s) = E^{\pi}\{R_{\mathrm{k}}|s_{\mathrm{k}} = s\} = E^{\pi}\{\sum_{n=0}^{N} \gamma^{n} r_{\mathrm{n+k+1}}|s_{\mathrm{k}} = s\} \tag{5-1}$$

Where E^{π} denotes the expected value, given that the agent follows the policy π [51]. Similarly, the action-value function is given by:

$$Q^{\pi}(s,a) = E^{\pi}\{R_{\mathrm{k}}|s_{\mathrm{k}} = s, a_{\mathrm{k}} = a\} = E^{\pi}\{\sum_{n=0}^{N} \gamma^{n} r_{\mathrm{n+k+1}}|s_{\mathrm{k}} = s, a_{\mathrm{k}} = a\} \tag{5-2}$$

The policy that maximizes the value functions can now be noted as the optimal policy, π^*. The optimal state-value and action-value functions, in state s, are denoted by $V^*(s)$ and $Q^*(s,a)$, respectively.

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{5-3}$$

and

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{5-4}$$

In RL, either the state- or the action-value function is used to define the optimal policy, π^*. Never are they used at the same time. If the value functions are known, it is the task of the agent to learn a policy that maximizes the state- or action-value function [51].

The policy is a mapping from states to actions ($s_{\mathrm{k}} \rightarrow a_{\mathrm{k}}$). Two policy types can be distinguished, a deterministic policy, $\pi_{\mathrm{k}}(s)$, and a stochastic policy, $\Pi_{\mathrm{k}}(s,a)$. Most often the policy is deterministic, and most straightforwardly the policy is to assign the action to each state that maximizes the action-value (Q-value). Only following the policy of taking the action that maximizes the action-value is called a greedy policy, if this policy is always adhered to an optimal policy will rarely be found. During learning, a non-greedy policy, called exploration, can be followed. This allows the exploration of new parts of the environment where potentially better solutions can be found [52]. The stochastic policy is a mapping from a state-action pair to a probability that this action will be chosen. A result of a stochastic policy is that in every iteration a different action can be chosen, while the mapping is not changed [51].

Q-Learning

One of the most well-known control algorithms in RL is Q-learning, a variation of Q-learning will be used in the remainder of this work. Q-learning is an off-policy temporal difference control algorithm that makes use of discrete MDP's. In it's simplest form it is defined as

$$Q(s_{\mathrm{k}}, a_{\mathrm{k}}) \leftarrow Q(s_{\mathrm{k}}, a_{\mathrm{k}}) + \alpha(r_{k+1} + \gamma \max_a Q(s_{\mathrm{k+1}}, a) - Q(s_{\mathrm{k}}, a_{\mathrm{k}})) \tag{5-5}$$

To guarantee convergence, all state-action pairs need to be updated continually [51]. The procedural form of the Q-learning algorithm is shown in Figure 5-1.

Initialize Q(s,a), $\forall s \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal - state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q
    Take action A, observe R, S'
    $Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha(r_{k+1} + \gamma \max_a Q(s_{k+1}, a) - Q(s_k, a_k))$
    $S \leftarrow S'$:
  until S is terminal

Figure 5-1: Q-learning: An off-policy temporal difference control algorithm [52].

5-2 Function approximation for reinforcement learning

Simplification of a realistic environment can deteriorate the performance of the controller of that environment. Searching a state space of large problems is infeasible because of data storage limitations and the need to encounter each state and action within the state-action space [52]. A method of dealing with the posed limitations on problems with large state-spaces is to approximate the entire state-space from experience from a subset of that state-space. RL algorithms can be extended to high dimensional problems by using function approximation. In this section a short introduction to function approximation will be given. An extension on this theory will be discussed in chapter 6.

MDP's are subject to the curse of dimensionality, the number of action-values that has to be stored in a table grows exponentially with the number of states and actions. This growth in stored action-values translates in a growth in the necessary storage capacity and computing power. Furthermore, when the state or action spaces include continuous variables or complex sensations, most states encountered will never have been experienced before [52]. The only way to learn anything on these tasks is to generalize from previously experienced states to ones that have never been seen. The number of states and actions in greenhouse climate control is large, function approximation can prove a viable method to mitigate the problems encountered in the presence of large state-action spaces.

Approximation methods use a parameter vector $\mathbf{w} \in \mathcal{R}^n$. Th approximated value of the state s given weight vector $\mathbf{w}$ can then be written $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$. The value $\hat{v}$ might be the function computed by an artificial neural network, with $\mathbf{w}$ the vector of the connection

weights. By changing the weights, a wide range of different functions $\hat{v}$ can be implemented by the network. A decision tree might also compute the function $\hat{v}$, where $\mathbf{w}$ is the vector defining all the split points and leaf values of the tree. The number of parameters n is usually much less than the number of states. Changing one parameter within the vector changes the estimated value of many states.

Many function approximation methods exist, these methods have been extensively researched and written about. It is outside the scope of this work to discuss all existing methods. Interested readers are referred to part 2 of [52] for further readings. In this work, in chapter 6, the use of MPC as a function approximation method is discussed.

Chapter 6

Reinforcement learning with economic NMPC

A powerful tool to perform optimal control without depending on a model of the system is reinforcement learning. RL methods are often based on learning the optimal control policy. Unfortunately, RL can't provide strong guarantees on the behavior of the resulting control scheme. Model predictive control (MPC), on the other hand, is a standard tool for the closed-loop control of systems with constraints and limitations. Furthermore, the theory behind (N)MPC has been developed thoroughly. In [53], it is shown that an (E)NMPC scheme can be tuned to deliver an optimal policy of the real system, even when using a wrong model. The model and (N)MPC parameters can be changed with the help of RL strategies to achieve better performance. This is seen as an opportunity in the case of greenhouse climate control and will be further explained in this chapter.

RL is a tool for tackling Markov decision processes without depending on the model underlying the state transitions. Most RL methods rely on observed state transitions and realisations of the stage cost to increase the performance of the control policy. Indirect learning methods learn an approximation of the optimal action-value function underlying the MDP. Generally, the action-value function is unknown a priori, typically a generic function, such as a deep neural network, is used to approximate it. Direct RL methods learn the optimal policy directly. Most direct methods are based on policy gradient methods. Direct RL methods often use DNN's to approximate the optimal policy and the (action-) value function associated. It can be difficult to formally analyze the closed loop behavior of systems subject to an approximate optimal policy supported by a DNN or a generic function approximation method [53], furthermore DNN's limit the amount of knowledge that can be included about the system and its underlying constraints. (E)NMPC schemes can be used instead of DNN's to support the parametrization to approximate the (action-) value functions and the policy [53]. By adapting the stage cost, terminal cost, and constraints the NMPC scheme can deliver the optimal control policy, even if the underlying model is incorrect. Using (E)NMPC as a parametrization for RL allows one to use the rich theory underlying (E)NMPC schemes in the context of RL, system knowledge and constraints can be included by using (E)NMPC as a

function approximator. Most of the theory in this chapter is based on [53] [54] [55] [56]. The reader is referred to these papers for a more in depth explanation of the theory and proofs thereof.

6-1 Using ENMPC as a function approximator

A parametrized ENMPC scheme can be used to approximate the optimal policy and the value functions $\pi_\star$, $V_\star$, and $Q_\star$, even if the underlying model of the ENMPC is not accurate. System knowledge and constraints can be captured in the ENMPC scheme, while the RL algorithm will help improve the performance of the system.

The true greenhouse model can be described by a discrete MDP having the state transition dynamics:

$$\mathbb{P}[s_+|s,a] \tag{6-1}$$

Which, in more typical control fashion, can also be written as:

$$s_+ \;=\; f(s,a,\zeta) \tag{6-2}$$

For this thesis it is assumed that the true model of the system is not known and an approximate model is taken into account for the control purposes, having the following transition dynamics:

$$\mathbb{P}[\hat{s}_+|s,a] \tag{6-3}$$

This model does not necessarily match the real model (Equation 6-1), indicated by $\hat{s}_+$. In the approximate instance, f is replaced by $\hat{f}_\theta$.

The ENMPC scheme will be based upon the imperfect model of the greenhouse system, $\hat{f}_\theta$, and seek the minimization of the associated stage cost $\hat{L}(s,a)$.

$$\hat{L}(s,a) \;= \begin{cases} Q_\star(s,a) - \gamma \mathcal{V}_\star^+(s,a), & \text{if } |\mathcal{V}_\star^+(s,a)| < \infty \\ \infty, & \text{otherwise.} \end{cases} \tag{6-4}$$

Where $\mathcal{V}^+(s,a) = \mathbb{E}[V_\star(\hat{s}_+|s,a]$, and the expectation is taken over the approximate system model. It should be noted that the conditional formulation of the stage cost is to create a well-defined $\hat{L}$. In [53], it is stated that, under some conditions, the optimal policy $\pi_\star$ that minimizes the stage cost for the true dynamics is also generated by using the approximate model with its associated stage cost $\hat{L}$. Thus, making it possible to find the optimal policy based on a wrong model. $\hat{L}$ can be learned directly from the data.

$\hat{L}$ is not known a priori and will have to be learned, as a result the parametrization of the mixed constraints will also have to be learned through RL tools. The constraints can be used in the ENMPC scheme to exclude the undesirable states and inputs. In the most generic form the constraints will consist of pure input constraints, and mixed constraints which constrain combinations of states and inputs:

$$g(a) \;\leq\; 0, \quad h(s,a) \;\leq\; 0 \tag{6-5}$$

Since RL can't handle infinite values easily the domain where $\hat{L}$ is finite needs to be captured, the pure input constraints can remain fixed, but the mixed constraints need to be modified. A slack variable is added to relax the mixed constraints as follows:

$$h_\theta(x_\mathrm{k}, u_\mathrm{k}) \leq \sigma_\mathrm{k},\ h^\mathrm{f}_\theta(x_\mathrm{N}) \leq \sigma_\mathrm{N}. \tag{6-6}$$

As a result, the parametrization of the value function $V_\star$ uses the following ENMPC scheme. The value function ENMPC scheme is parametrized by θ.

$$\begin{aligned} V_\theta(s) = \min_{\mathrm{u,x},\sigma}\ & \lambda(x_0) + \gamma^\mathrm{N}(V^f_\theta(x_N) + w_\mathrm{f}^\top \sigma_\mathrm{N}) + \sum_{\mathrm{k}=0}^{\mathrm{N}-1} \gamma^\mathrm{k}(l_\theta(x_\mathrm{k}, u_\mathrm{k}) + w^\top \sigma_\mathrm{k}) \\ s.t.\ & x_{\mathrm{k}+1} = f_\theta(x_\mathrm{k}, u_\mathrm{k}),\ x_0 = s \\ & g(u_\mathrm{k}) \leq 0, \\ & h_\theta(x_\mathrm{k}, u_\mathrm{k}) \leq \sigma_\mathrm{k},\ h^\mathrm{f}_\theta(x_\mathrm{N}) \leq \sigma_\mathrm{N}. \end{aligned} \tag{6-7}$$

The ENMPC scheme holds parametrizations of the model, f_θ, the mixed constraints, h_θ, the initial cost, λ_θ, the stage cost, l_θ, and the terminal cost, V^f_θ.

RL is used to increase the closed-loop performance of the system, by modifying the parameter values for the parametrization of the model, constraints and costs. The ENMPC scheme above is a classic ENMPC formulation in the case the $\gamma = 1$ and $\lambda_\theta = 0$ [50]. The mixed constraints are relaxed using slack variables, σ_k. The l_1 constraints relaxation is critical for the TD RL approaches that will be used in this work. If this relaxation were absent, the value functions might take on infinite values. Only finite values can be used for the TD learning, infinite values would cause the learning to be meaningless/infinite. The weights placed on the slack variables are denoted by w, w_f. The initial cost, λ_θ, is used to form nominal stability guarantees of the ENMPC scheme.

The action that is taken by the system is defined as taking the first element of the input sequence generated by solving the value function ENMPC scheme for a given state.

$$a = u_0^\star \tag{6-8}$$

Accordingly, the action-value function, Q_θ of the Q-Learning problem is given by:

$$\begin{aligned} Q_\theta(s) = \min_{\mathrm{u,x},\sigma}\ & \lambda(x_0) + \gamma^\mathrm{N}(V^f_\theta(x_\mathrm{N}) + w_\mathrm{f}^\top \sigma_\mathrm{N}) + \sum_{\mathrm{k}=0}^{\mathrm{N}-1} \gamma^\mathrm{k}(l_\theta(x_\mathrm{k}, u_\mathrm{k}) + w^\top \sigma_\mathrm{k}) \\ s.t.\ & x_{\mathrm{k}+1} = f_\theta(x_\mathrm{k}, u_\mathrm{k}),\ x_0 = s \\ & g(u_\mathrm{k}) \leq 0, \\ & h_\theta(x_\mathrm{k}, u_\mathrm{k}) \leq \sigma_\mathrm{k},\ h^f_\theta(x_\mathrm{N}) \leq \sigma_\mathrm{N}. \\ & u_0 = a. \end{aligned} \tag{6-9}$$

In which only the final equality constraint is added to the scheme of Equation 6-7. A further difference between the two schemes, is that the cost function for the value function scheme can be slightly perturbed in order to enhance exploration. This perturbation happens by the addition of a value resulting from $f(\mathbf{u}_0, \mathbf{q}) := \rho||(\mathbf{u}_0 - \mathbf{q})||$, or $\mathbf{q}^T\mathbf{u}_0$, where $\mathbf{q}$ is randomly

chosen. This parametrization will automatically satisfy the fundamental equalities underlying the Bellman equations:

$$\pi_\theta(s) = arg\min_{\mathrm{a}} Q_\theta(s,a) \tag{6-10}$$

$$V_\theta(s) = \min_{\mathrm{a}} Q_\theta(s,a) \tag{6-11}$$

6-2 RL for ENMPC

In theory, it is possible to generate the optimal policy and value functions using an ENMPC scheme based on an inaccurate model. In practice, a smart parametrization of the ENMPC scheme should be relied upon. The goal is to adjust the parameters θ in such a way that the policy resulting from the ENMPC scheme fits the optimal policy as closely as possible. The adjustment of the parameters relies on RL algortithms. In this work, classical RL approaches will be used, which are mostly gradient based. Hence, these methods require the gradient of the value functions, which requires the sensitivities of the optimal values of Q_θ [57]. The sensitivity analysis of optimization problems is detailed in [58] and delivers $\nabla_\theta Q_\theta$, needed in Equation 6-16. In order to evaluate the gradients of the functions Q_θ, V_θ and π_θ the Lagrange function associated with the action-value function ENMPC problem has to be found. It is observed that:

$$\nabla_\theta Q_\theta = \nabla_\theta \mathcal{L}_\theta(s, y^\star) \tag{6-12}$$

Where $y^\star$ holds the primal-dual variables associated to the ENMPC problem Equation 6-9. The Lagrangian of the action-value function is as:

$$\begin{aligned}\mathcal{L}(s,y) =& \lambda_\theta(x_0) + \gamma^{\mathrm{N}} V_\theta^{\mathrm{f}}(x_{\mathrm{N}}) + \chi_0^\top (x_o - s) + \mu_{\mathrm{N}}^\top h_\theta^{\mathrm{f}}(x_{\mathrm{N}}) \\ &+ \sum_{\mathrm{k}=0}^{\mathrm{N}-1} \chi_{\mathrm{k}+1}^\top (f_\theta(x_{\mathrm{k}}, u_{\mathrm{k}}) - x_{\mathrm{k}+1}) + \nu_{\mathrm{k}}^\top g_\theta(u_{\mathrm{k}}) \\ &+ \gamma^{\mathrm{k}} l_\theta(x_{\mathrm{k}}, u_{\mathrm{k}}) + \mu_{\mathrm{k}}^\top h_\theta(x_{\mathrm{k}}, u_{\mathrm{k}}) + \zeta^\top (u_0 - a)\end{aligned} \tag{6-13}$$

Where χ, μ, ν, and ζ are the multipliers associated to the constraints of Equation 6-9. The gradient of Q is then created by differentiating the Lagrangian with respect to the parameters of interest.

6-3 Q-learning for ENMPC

Q-learning has been introduced in section 5-1. One approach to Q-learning is based on parameter updates driven by the temporal difference [52]. The TD error is calculated as:

$$\tau_{\mathrm{k}} = L_\theta(s_{\mathrm{k}}, a_{\mathrm{k}}) + \gamma V_\theta(s_{\mathrm{k}+1}) - Q_\theta(s_k, a_{\mathrm{k}}) \tag{6-14}$$

with gamma as the discount rate. L_θ represents the rewards function:

$$L_\theta(s_{\mathrm{k}}, a_{\mathrm{k}}) = l_\theta(x_{\mathrm{k}}, a_{\mathrm{k}}) + w^\top max(0, h_\theta(x_{\mathrm{k}}, a_{\mathrm{k}})) \tag{6-15}$$

Where l_θ is based on the inputs and the states, and $w^\top max(0, h_\theta(x_{\mathrm{k}}, a_{\mathrm{k}}))$ is the importance placed on the constraint violations.

The parameter update is then given by

$$\theta \leftarrow \theta + \alpha \tau_{\mathrm{k}} \nabla_{\theta} Q_{\theta}(s_{\mathrm{k}}, a_{\mathrm{k}}) \tag{6-16}$$

in which $\nabla_{\theta} Q_{\theta}(s_{\mathrm{k}}, a_{\mathrm{k}})$ is found by differentiating the Lagrangian associated with the ENMPC problem of the action-value function with respect to the parameters, as explained above. The action, a_{k}, is selected according to the ENMPC policy. The scalar α is the step size, determining the speed at which θ is altered.

Batch updates can be used in the case of episodic tasks, such as the greenhouse climate control problem. Now, learning is performed with an alternative ENMMPC scheme. The policy is based on the original ENMPC scheme, of which the parameters can be replaced with the learned parameters at convenience. Instead of updating the parameters every time-step, the update happens at convenience. The information contained in all time-steps between parameters can be combined and used for the next parameter update. In the case of episodic tasks, the system behavior might change during an episode and system updates based on partial information can deteriorate performance.

In the remainder of this thesis Q-learning for ENMPC will be referred to as Q-ENMPC.

Chapter 7

Set-up

To test the performance of using Q-ENMPC, simulations will be performed in the environment as described in chapter 3. Matlab will be used for the modeling and simulation of these experiments . Before any simulations can be run of Q-ENMPC, the MPC structure needs to be tested and verified. Secondly, parameters need to be selected which the system can alter to improve performance. The MPC system will be verified in section 7-1 and the parameters are selected in section 7-2. Lastly, performance indicators are presented in section 7-3.

7-1 Verification of the results with only the MPC controller

The goal of this study is to improve the performance on the problem laid out in chapter 3. Previous results were obtained in [49]. The goal of this section is to verify that the results in [49] are reproducible. In order to verify the results from [49], the MPC from the paper is recreated and experiments are run. This is, in theory, easy to do and beneficial for later stages of this research as the MPC controller itself forms the basis for the function approximator that is used for the RL agent.

Results

The results of [49] can be seen in Table 7-1. From these results it is clear that increasing the parameter uncertainty, $\Sigma_{\hat{p}}$, results in a decrease of crop dry weight at harvest time.

The results from replicating the experiments are shown in Table 7-2. It has been chosen not to verify the results at every uncertainty interval. Running an experiment takes about a day of computation time and these results are deemed sufficient to verify that the results of [49] can be replicated. Although the results are not identical, they are assessed as being similar enough.

The origin of the discrepancy between the verification results in Table 7-2 and the results from [49] has been identified as the amount of samples taken. The amount of samples for the

$\Sigma_{\hat{p}}$	$\delta W\%$	$\delta\text{rms}(u_1)$ %	$\delta\text{rms}(u_2)$ %	$\delta\text{rms}(u_3)$ %
$0.05I$	-3.9	-14.8	36.7	16.8
$0.1I$	-6.6	-30.9	63.1	37.6
$0.15I$	-8.9	-49.8	85.8	66.9
$0.2I$	-11.2	-79.7	96.2	90.4

Table 7-1: The performance of the MPC controller as described in [49]. From the results it is clear that an increase in parameter results in a decrease in performance. Increasing uncertainty results in lower crop dry weight at harvest time and increased energy inputs. In the paper, the number of samples that is taken into account at each time-step, Ns, is 20.

$\Sigma_{\hat{p}}$	$\delta W\%$	$\delta\text{rms}(u_1)$ %	$\delta\text{rms}(u_2)$ %	$\delta\text{rms}(u_3)$ %
$0.1I$	-5.4	-29	55	33
$0.2I$	-9.6	-69	92	76

Table 7-2: The performance of the MPC controller during the verification experiments. From the results it is clear that an increase in parameter results in a decrease in performance. Increasing uncertainty results in lower crop dry weight at harvest time and increased energy inputs. For these results Ns was taken to be 10.

verification of the results was taken to be 10, while [49] uses 20 samples at each time step. The result of taking 20 samples can be seen in Table 7-3. The results in Table 7-3 are closer to the results obtained in [49] than the results in Table 7-2. It is determined that the results are similar and the newly created MPC scheme can be used for the remaining research.

$\Sigma_{\hat{p}}$	$\delta W\%$	$\delta\text{rms}(u_1)$ %	$\delta\text{rms}(u_2)$ %	$\delta\text{rms}(u_3)$ %
$0.2I$	-10.9	-80.6	98.2	90.6

Table 7-3: The performance of the MPC controller during the verification experiments. From the results it is clear that an increase in parameter results in a decrease in performance. Increasing uncertainty results in lower crop dry weight at harvest time and increased energy inputs. For these results Ns was taken to be 20.

7-2 Parameter selection

The model and the MPC problem are described by a number of parameters. Adjusting all the parameters of the model and the various parameters for the MPC implementation is likely too time intensive and not always necessary to achieve significantly improved performance. This section will describe the process of finding the parameters that promise to have the greatest influence on the performance of the controller.

Parameters that are the most sensitive are the most important to find the correct value for, as they influence the performance of the controller the most. Insensitive parameters are of less importance as they do not influence the performance of the controller as much.

Three sources of information are used for the parameter selection. Firstly, various published sensitivity analyses are studied to find the most important parameters of the model describing the dynamics of the system itself. Secondly, a new sensitivity analysis is done to verify the

results found in the papers. Lastly, [54] is used to identify the most important parameters of the Q-ENMPC scheme.

First, the model parameters that will be subject to change by the Q-ENMPC algorithm will be selected in subsection 7-2-1. Then, the MPC control problem parameters will be selected in subsection 7-2-2.

7-2-1 Model parameter selection

The selection of the most important model parameters is done with help from [31], [29] and [59]. The model, as described in chapter 3, consists of 22 parameters. The question that will be answered in this section is which parameters impact the performance of the model the most in the face of uncertainty.

In [29] a sensitivity analysis is presented that emphasizes the time evolution of the model sensitivity. A first order method is used, the sensitivity of the state trajectories to small parameter deviations around a nominal value is evaluated by simultaneously solving the sensitivity equations and the dynamic model. The model that is used in [29] is not identical to the model that is used in the present study, a two-state variable model is used that separates the structural and non-structural dry-weight. The present study uses a single state variable that sums the total crop dry-weight.

The results of this analysis indicate that the most sensitive parameters are the maximum growth rate, the yield factor, the light use efficiency and the carboxylation constant with respect to temperature. The leaf area index, the Q_{10}-factor for growth, and the extinction coefficient also have a large effect.

In [59] the relative sensitivity of the model to changes in parameters was evaluated using the sensitivity function

$$S_{\mathrm{i}} = \frac{\partial Y_{\mathrm{dw}}(T)}{\partial p_{\mathrm{i}}} \frac{p_{\mathrm{i}}}{Y_{\mathrm{dw}}(T)} \tag{7-1}$$

where S_{i} is the value of the sensitivity function for a variation of parameter i, $Y_{\mathrm{dw}}(T)$ is the simulated dry weight at harvest time T using the original parameters, p_{i} is the original parameter value, ∂p_{i} is a small variation in parameter i while keeping the other parameters constant and $\partial Y_{\mathrm{dw}}(T)$ is the difference between the simulated dry weight at harvest time with and without the parameter variation. If $S_{\mathrm{i}} = 1$, a given fractional change in the value of the parameter i produces the same fractional change in the yield, $Y_{\mathrm{dw}}(T)$.

The results from [59] indicate that the yield factor and the light use efficiency play a major role in the determination of lettuce crop growth. The parameter determining the carboxylation conductance, and the extinction coefficient and the leaf area ratio affect to a smaller extend the simulatied dry weight at harvest time.

This same equation can be applied to the model used in the present paper. Testing per parameter what the influence on the model outcome is of a 10% increase in parameter value, compared to a run with nominal parameter values. The found relative sensitivities are shown in Table 7-4. Table 7-5 shows the relative sensitivity for a 10% decrease in parameter value and Table 7-6 shows the relative sensitivity for a 5% increase in parameter value.

Parameter	Relative Sensitivity
ventilation leakage through the cover	−0.0104
CO2 capacity of the greenhouse	−0.0170
Vapor capacity of the greenhouse	0.0083
effective heat capacity of the greenhouse air	−0.0101
heat capacity per volume of greenhouse air	0.0205
overall heat transfer through the cover	0.0027
heat load coefficient due to solar radiation	0.0128
yield factor	**0.4595**
respiration rate	−0.0187
respiration coefficient	0.0001
effective canopy surface	**0.4008**
light use efficiency	**0.3782**
temperature influence on photosynthesis	−0.0698
temperature influence on photosynthesis	**0.1379**
temperature influence on photosynthesis	−0.0287
carbon dioxide compensation point	−0.0021
coefficient of leaf-air vapor flow	−0.0167

Table 7-4: The relative sensitivity of the model parameters of the model used in the present paper for a 10% increase in parameter value. In bold the parameters with the largest influence are shown.

The results presented in Table 7-4 indicate that three parameters play a major role in determining the lettuce crop growth, namely the yield factor, the light use efficiency, and the effective canopy surface. To a lesser extend the temperature influence on photosynthesis affect the crop weight at harvest time.

It can be concluded, both from the existing parameter sensitivity studies and from the parameter sensitivity study conducted in the present paper that the model parameters that have the greatest influence on the yield of the system are the yield factor, light use efficiency, effective canopy surface, and to a lesser extend the temperature influence on the photosynthesis. The relative sensitivities, at a parameter deviation of 5%, are 0.4786, 0.4185, 0.4011, and 0.1496, respectively. In [31] it is concluded that the vapour pressures, perturbation constants on the outside CO_2 concentration, humidity and solar radiation, gas constant, and conversion constant between Kelvin and Celsius also have a major influence. It is argued in the present study that these parameters are well known and fixed.

The performance of the controller is not only dependent on the yield of the system, as will be discussed in section 7-3, the total inputs to the system during a growth cycle also play a major role to determine the performance. The parameters that should be changed to have an influence on the control actions by the controller are not necessarily the same as those that influence the yield of the system. For this study the results of the sensitivity study are combined with the model equations. The model parameters that have the greatest influence on the states of the system are determined to be the CO_2 capacity of the greenhouse, heat capacity of the greenhouse air and coefficient of leaf-air vapor flow, for the CO_2 concentration, temperature and humidity, respectively.

It is thus decided that the parameters that will be subject to change by the RL agent are

Parameter	Relative Sensitivity
ventilation leakage through the cover	−0.0003
CO2 capacity of the greenhouse	−0.0136
Vapor capacity of the greenhouse	−0.0119
effective heat capacity of the greenhouse air	−0.0180
heat capacity per volume of greenhouse air	0.0425
overall heat transfer through the cover	−0.0331
heat load coefficient due to solar radiation	0.0001
yield factor	**0.4983**
respiration rate	−0.0410
respiration coefficient	0.0001
effective canopy surface	**0.4375**
light use efficiency	**0.4162**
temperature influence on photosynthesis	−0.0556
temperature influence on photosynthesis	**0.2121**
temperature influence on photosynthesis	−0.0275
carbon dioxide compensation point	−0.0021
coefficient of leaf-air vapor flow	−0.0167

Table 7-5: The relative sensitivity of the model parameters of the model used in the present paper for a 10% decrease in parameter value. In bold the parameters with the largest influence are shown.

Parameter	Relative Sensitivity
ventilation leakage through the cover	−0.0002
CO2 capacity of the greenhouse	−0.0092
Vapor capacity of the greenhouse	0.0005
effective heat capacity of the greenhouse air	−0.0101
heat capacity per volume of greenhouse air	0.0182
overall heat transfer through the cover	−0.0032
heat load coefficient due to solar radiation	0.0131
yield factor	**0.4786**
respiration rate	−0.0187
respiration coefficient	0.0001
effective canopy surface	**0.4011**
light use efficiency	**0.4185**
temperature influence on photosynthesis	−0.0807
temperature influence on photosynthesis	**0.1496**
temperature influence on photosynthesis	−0.0285
carbon dioxide compensation point	−0.0024
coefficient of leaf-air vapor flow	−0.0167

Table 7-6: The relative sensitivity of the model parameters of the model used in the present paper for a 5% increase in parameter value. In bold the parameters with the largest influence are shown.

the yield factor, the CO_2 capacity of the greenhouse, heat capacity of the greenhouse air and coefficient of leaf-air vapor flow.

7-2-2 MPC control problem parameter selection

In this section the MPC controller parameters that will be subject to change by the RL agent will be identified. The main source of information to idententify the parameters is [54], which is the main reference for the overall control method.

The important tuning parameters for the control problem are the control horizon, prediction horizon, cost function with initial, stage and terminal costs, and constraints.

For this study it is desirable to keep the control problem as similar to the control problem from [49] as possible. It is thus decided not to alter the horizons. Furthermore, [54] shows that the Q-ENMPC scheme can deliver the optimal control policy, even if the underlying model is incorrect, by adapting the stage cost, terminal cost, and constraints only. A simple conclusion is thus to only include these as parameters in the RL scheme. The stage and terminal costs are quite logical, while the constraints may not be. The constraints can be used in the ENMPC scheme to explicitly exclude undesirable states and inputs. The pure input constraints are arguably fixed, but the mixed constraints should be modified to capture the domain where the cost is finite.

Adjustable parameters are added for the stage cost of the three inputs and the yield. Furthermore, a parameter is added for the terminal yield and a single parameter is added as the initial cost. Finally, an adjustable parameter is added for the weighting on the constraint violations.

The total amount of parameters that can then be adjusted by the RL algorithm is 11. Of these, 7 parameters are newly added parameters to adjust the cost function, while four parameters are existing model parameters.

7-3 Performance indicators

The first performance indicator used in this work was introduced in [60], an economic profit indicator. The economic profit indicator calculates the profit that stems from a certain growth cycle of lettuce, when the lettuce is harvested and sold at the end of that cycle. The function is as follows:

$$EPI = \phi(y_1(t_\mathrm{f})) - \sum_{t_\mathrm{b}}^{t_\mathrm{f}} (c_\mathrm{q} u_\mathrm{q}(t) + c_{\mathrm{CO}_2} u_{\mathrm{CO}_2}(t))dt \tag{7-2}$$

where $\phi(y_1(t_\mathrm{f}))$ is the gross income from selling the harvested lettuce, which is obtained at harvest time t_f, in $Hflm^{-2}$. The operating cost of the airconditioning equipment is indicated by $c_\mathrm{q} u_\mathrm{q}(t) + c_{\mathrm{CO}_2} u_{\mathrm{CO}_2}(t)$. The auction price of the lettuce is assumed to follow a linear ratio $\phi(y_1(t_\mathrm{f})) = c_{\mathrm{pri},1} + c_{\mathrm{pri},2} y_1(t_\mathrm{f})$ between the auction price and harvest weight. $c_{\mathrm{pri},1}$ in $Hflm^{-2}$ and $c_{\mathrm{pri},2}$ in $Hflkg^{-1}m^{-2}$. It is assumed that the operating cost of the air conditioning equipment are related to the amount of energy, u_q in (Wm^{-2}), and the amount of carbon dioxide introduced into the system, u_{CO_2} in $(kgm^{-2}s^{-1}$, linearly. The operating costs are

parameter	value
c_{co_2}	$42 \times 10^{-2}\ Hflkg^{-1}$
c_{q}	$6.35 \times 10^{-9}\ HflJ^{-1}$
$c_{\text{pri,1}}$	$1.8\ Hflm^{-2}$
$c_{\text{pri,2}}$	$16\ Hflm^{-1}$

Table 7-7: Parameters of the economic profit function.

parameterised by the price of energy, c_{q} in $(HflJ^{-1})$, and the price of carbon dioxide, c_{CO_2} in $(Hflkg^{-1})$. The values of these parameters can be found in Table 7-7.

The second performance indicator is the constraint violations during the growth cycle. The constraint violations are calculated as the total amount of constraint violation summed over the entire horizon. For example, the humidity constraint is always set at 70%. If, during three time-steps the humidity is at 75%, the total constraint violation is 15. During those same time-steps it could be that the temperature constraint is also violated by one degree, then the total constraint violation is 15, for the humidity violation, plus 3, for the temperature violation, for a total of 18. The total constraint violations over the growth cycle are divided by 50 to reach a more intuitive number.

Chapter 8

Case study: Q-learning for ENMPC

Simulations will be performed to compare the performance of the controller before and after learning the parameters. Performance will also be compared with the performance achieved with robust control and in the nominal case. The open-source software CasADI and solver IPOPT are used in a Matlab environment to solve the optimization problems, following a multiple-shooting approach with warm start.

The model that will be used in the simulations has been thoroughly discussed in chapter 3. The main theory behind this work has been presented in chapter 6. The set-up, including parameter selection, verification of the MPC controller and explanation of the performance indicators has been discussed in chapter 7.

Three case studies will be conducted to compare performance between the Q-ENMPC, robust MPC and nominal MPC. In the first case study, emphasis will be placed on the performance based on the economic profit function, the second case study will focus on minimizing constraint violations, and the third case study will combine both.

The remainder of the chapter will first detail the practical implementation, including the changes made to the ENMPC scheme with respect to [49] and the considerations for the RL scheme. Secondly, the case studies will be explained and the results will be presented.

8-1 Practical implementation

8-1-1 The ENMPC scheme

The basis of the ENMPC scheme that is used in this work is the same as the one used in [49]. Some adjustments are made to the cost function and constraints, with respect to their work. In this section the changes in the ENMPC scheme of this work with respect to [49] will be pointed out.

Cost function

The cost function for the action-value function is constructed by including a cost on all the inputs of the system and rewarding the yield. The cost function is implemented as follows:

$$\begin{aligned} J(s) =& \lambda_\theta \\ &+ \sum_{k=0}^{N-1} \sum_{i=1}^{3} c_{u,i} * u_{i,k} \\ &+ \sum_{k=1}^{N-1} c_{y,1} * (y_k - y_{k-1}) \\ &+ c_{y,2} * y_N \\ &+ \sum_{k=1}^{N} w * \sigma_k \end{aligned} \tag{8-1}$$

The initial cost, λ_θ, is initialised at 100. The inputs are penalised per time-step in the prediction-horizon. The three different inputs are weighted differently by $c_{u,i}$. The yield is weighted via $c_{y,1}$ and $c_{y,2}$. Lastly, the slack variables, which register constraint violations, are weighted by w. The value of the weights $c_{u,i}$, $c_{y,i}$ and w will be learned. Furthermore, to reward an increase in yield, the yield related weighting is initialized negative. Via the third term in Equation 8-1, part of the cost on the yield is placed on the terminal yield within the time horizon, weight is also placed on the increase (or decrease) per time-step in the prediction horizon, via the second term in Equation 8-1.

Exploration of the system can be enhanced by using a perturbation in the cost function used for approximating the value function:

$$V_\theta(s) = Q_\theta(s) + \mathbf{q}^{\mathrm{T}}\mathbf{u}_0 \tag{8-2}$$

Where $\mathbf{q}$ is a weight placed on the first input in the prediction horizon. This weight has a uniform distribution with mean 0. With $\mathbf{q}$ the exploration is regulated. It has been concluded that exploration is not beneficial for this research, as such $\mathbf{q}$ is set to zero. A short discussion on why $\mathbf{q}$ is set to zero in this thesis work is included in Appendix A.

Constraints

The constraints follow the structure of the model discussed in chapter 3. It should be noted that the inequality constraints are extended with slack variables and an extra constraint on the action-value function that is shown in Equation 6-9. The first input of the action-value function, Q_θ, is fixed to the first input resulting from calculating the value-function, V_θ.

The objective of the final optimization problem is given by Equation 8-1. The constraints are given in Equation 3-25. Slack variables are added to the inequality constraints, similarly to Equation 6-9. A final constraint is added with the addition of the final equality constraint of Equation 6-9.

Lagrangian

An important step in the Q-learning approach is evaluating the gradients of the functions V_θ, Q_θ and π_θ. This evaluation can be done with the help of the Lagrangian function associated with the ENMPC scheme. The RL update and ENMPC scheme are explained in further detail in chapter 6.

The Jacobian and the Hessian of the Lagrangian can easily be found using Matlab. The Lagrangian is first constructed symbolically. The numerical values for the Lagrange multipliers and constraints are a results of the optimization step, which can be substituted into the symbolic Lagrangian. A numerical value for the Lagrangian and its derivatives is then found.

8-1-2 Changing parameters

For the various case studies, a couple of factors are of importance to make the system learn correctly. In the following, these factors will be discussed briefly.

RL reward function

The reward function of the RL agent is the main differentiator in the learning process. It dictates what is good progress and what is bad behavior of the system. The main parts that will be present in the reward function are a rewards for the performance, and a penalty for constraint violations.

c_1 and c_2 are the weights placed on the performance and the constraint violations, allowing one to decide what is important in the learning process. Generally, the value of c_1 will be between zero and a negative integer, while the value of c_2 will be between 0 and a positive integer. The reward function is part of the TD error as described in section 6-3. Hence the reward function is defined as:

$$L_\theta(s_k, a_k) = c_1 * l_\theta(x, a) + c_2 * \sum_{k=t_\mathrm{b}}^{t_\mathrm{f}} (\max\{0, h_\theta(x_k, a_k)\}) \tag{8-3}$$

Where $l_\theta(x, a)$ is taken as the EPI, explained in section 7-3, minus a fixed integer, 1.8. $c_2 * \sum_{k=t_\mathrm{b}}^{t_\mathrm{f}}(\max\{0, h_\theta(x_k, a_k)\})$ is the sum of the magnitude of the constraint violations at each time-step, as discussed in section 7-3.

Exploration

Exploration is done through the addition of a perturbation to the value function in the ENMPC scheme. This perturbation is explained in further detail in section 6-1. In this work, no exploration will be included.

Learning rate

The learning rate determines the rate at which the parameter values are adjusted during learning. High learning rates can cause the system to 'jump' around the optimal values or even cause instability, while too low of a learning rate causes slower learning than necessary.

The learning rates are initiated to cause a 1% change in the values of the model and ENMPC parameters. The learning rate for the value of weight on the slack variables is higher, at a rate of 50%. The learning rates are decreased by 10% every episode during learning.

Slack variables

Slack variables are needed for Q-ENMPC to work, as described in chapter 6. The weights placed on the slack variables are an important factor in determining how conservative the system is and in decreasing the constraint violations. The initialization of the slack variables is a factor in the initial constraint violations.

Since the learning rate is a percentage of the initial value of the parameters, the weight on the slack variables should not be too low, as this could prohibit relevant learning of the parameter. A value of 5000 is settled upon as good starting ground for learning, while not yet prohibiting constraint violations so much that none are present at the start.

Parameter initialization

Multiple different initializations for the values of the model parameters are used in the case studies. First, the nominal values for the model parameters can be used, the values are given in Table 3-1. When the values for the model parameters are initialized with their nominal value, it will be referred to as nominal initialization.

Second, in some case studies the parameters are initialised within a band around the nominal parameters of 20%, as explained in section 3-2. This means that the value for each parameter deviates a maximum of 20% from the nominal value. The actual deviation is chosen at random, such that a significant increase in constraint violations is present, compared to the nominal parameters. When the values for the model parameters are initialized with this random value, it will be referred to as randomized initialization.

The model parameters values with which the system is initialised in the case of randomized initialization are presented in Table 8-1, replacing the nominal parameter values of Table 3-1.

parameter	value	parameter	value	parameter	value	parameter	value
$c_{1,1}$	0.37354	$c_{2,1}$	3.459	$c_{3,1}$	$2.52 \cdot 10^{4}$	$c_{4,1}$	4.799
$c_{1,2}$	$2.242 \cdot 10^{-7}$	$c_{2,2}$	$4.434 \cdot 10^{-7}$	$c_{3,2}$	1070	$c_{4,2}$	0.0032
$c_{1,3}$	50.69	$c_{2,3}$	$7.5 \cdot 10^{-6}$	$c_{3,3}$	6.03	$c_{4,3}$	9348
$c_{1,4}$	$2.07 \cdot 10^{-9}$	$c_{2,4}$	75560	$c_{3,4}$	0.207	$c_{4,4}$	8314
$c_{1,5}$	$4.52 \cdot 10^{-6}$	$c_{2,5}$	273.15			$c_{4,5}$	273.15
$c_{1,6}$	$3.65 \cdot 10^{-4}$	$c_{2,6}$	101325			$c_{4,6}$	17.4
$c_{1,7}$	$5.96 \cdot 10^{-4}$	$c_{2,7}$	0.044			$c_{4,7}$	239
$c_{1,8}$	$5.2 \cdot 10^{-5}$					$c_{4,8}$	17.269
						$c_{4,9}$	238.3

Table 8-1: Initialisation values of the model parameters in case of randomized initialization.

Thirdly, an initialization is used in which the value of all model parameters is initialized at 1.1 times their nominal value. When the values for the model parameters are initialized in this way, it will be referred to as initialization 1.1.

Lastly, a more complicated initialization is used. Initialization 1.1 is used as a starting point. This initialization is then used to learn for constraint violations, the reward function and results are shown in case study 2. By learning for constraint violations, the EPI decreases rapidly within a limited number of episodes. The parameters of the system in the third episode are used for the this initialization. This initialization will only be used in case study 1, it will be referred to as initialization 3. The benefit of this initialization is that it is known beforehand that within a limited amount of parameter updates a significantly increased EPI can be achieved.

Five parameters are usually kept to their nominal value, this is done because they represent well known physical constants (e.g. conversion factor from Kelvin to Celsius). In one cases, while using a randomized initialization these values are also randomized, in that case the initialization will be referred to as randomized$_\mathrm{r}$.

The initial value of the learn-able parameters of the ENMPC scheme are presented in Table 8-2. In the table, it can be seen that the inputs are penalised, while the yield is rewarded. The initial value of the learn-able parameters of the ENMPC scheme is the same for all case studies.

Parameter	Initial value
Initial cost	100
cost on CO_2 input	50
cost on ventilation	1
cost on heating input	1000
cost on yield increase	-10
cost on terminal yield	-100
Weight on slack variables	5000

Table 8-2: Initialisation parameters of the ENMPC scheme.

RL updates

During training the parameters selected in section 7-2 are updated by the RL agent. The interval between updates is a hyperparatemer in the controller design. In this study the parameters will be updated once per episode, with an episode being the length of one growth cycle. It is chosen to only update once per episode because the gradient of the system changes significantly during the episode. It would not be feasible to implement a changing learning rate. Due to there only being one update per episode, it is the amount of episodes that determines the amount of parameter updates.

The information contained in all time-steps is combined to form the TD error for the parameter update. As discussed in chapter 6, the TD error for each episode is given by

$$\tau_k = L_\theta(s_k, a_k) + \gamma V_\theta(s_{k+1}) - Q_\theta(s_k, a_k) \tag{8-4}$$

with the rewards function defined in Equation 8-3. As explained in section 6-3, the parameter update is then given by

$$\theta \leftarrow \theta + \alpha \tau_k \nabla_\theta Q_\theta(s_k, a_k) \tag{8-5}$$

Initialization	EPI	Constraint violations
Nominal	2.54	209
Randomized	2.73	296
1.1	2.89	504
Robust MPC	2.1	23

Table 8-3: Different initializations and their starting performance in terms of EPI and constraint violations.

The reward function is calculated once per episode, and changes for each case study. The gradient is averaged over the entire episode for each parameter. The average gradient is then used in the update equation above.

8-2 Starting performance

Before starting the case studies it is necessary to know the performance of the system with nominal parameters. Simulating with the nominal parameter values and a weight on the slack variables of $5e3$ results in an EPI of 2.54 ($Hflm^{-2}$), similar to the results of [49], and constraint violations of 209. Simulation with the randomized parameter initialization, resulted in an EPI of 2.73 ($Hflm^{-2}$), 7.5% higher than with nominal initialization, but the constraint violations are at 296, 42% higher than with the nominal parameters. Initialization 1.1 results in an EPI of 2.89 ($Hflm^{-2}$), 13.7% higher than in the nominal case. Constraint violations are at 504, 141% higher than in the nominal case.

One of the reasons for the constraint violations while having used the nominal parameters is the implementation of slack variables. These allow for a trade-off between performance and constraint violations. The slack variables are needed for the learning algorithm.

The performance achieved with robust control is significantly different than the performance achieved with nominal MPC. When sample-based robust MPC is used with 10 samples and an uncertainty band of 10% the EPI is 2.1 ($Hflm^{-2}$) and the constraint violations are 23. By increasing the amount of samples and the uncertainty band the performance decreases, with 20 samples and 20% uncertainty the EPI is 1.25 ($Hflm^{-2}$) and the constraint violations are 42. In this thesis the results of using robust MPC with 10 samples and a uncertainty band of 10% will be used for comparison.

8-3 Case study 1: Improvement of EPI from perturbed initial parameters

In this case study the performance of the Q-ENMPC will be compared to the MPC from [49] in terms of the achieved EPI. The main objective is to learn parameters and adjust them in such a way the EPI is increased, preferably such that the final performance that is achieved is better than by the controller with nominal initialization. Emphasis will be placed on the EPI in the RL reward function. The controllers will be initialized with the perturbed model parameter values and compared to a MPC with nominal model parameter values, both without exploration.

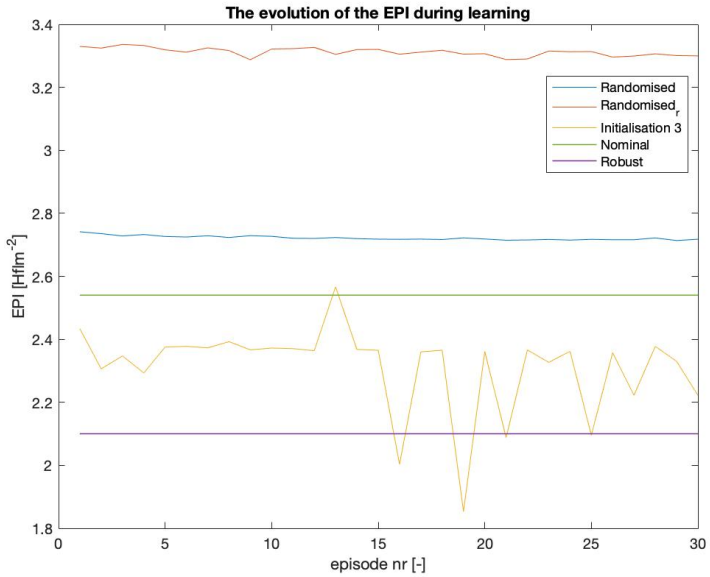

Figure 8-1: The evolution of the EPI during learning for the economic performance. The model parameters are initialized with the randomized initialization, in blue, randomized$_{\mathrm{r}}$ initialization, in orange, initialization 3, in yellow, nominal initialization, in green, and nominal initialization with robust MPC, in purple.

The RL reward function will focus only on the EPI in this case. The newly introduced RL agent with MPC controller as a function approximator will be used. The agent will learn for 30 episodes.

The RL reward function is constructed with c_1 being -200 in all instances. c_2 is set to zero, resulting in the following reward function:

$$\text{RL reward} = -200 * \left(c_{\text{pri,s}} y_1(t_{\text{f}}) - \sum_{k=t_{\text{b}}}^{t_{\text{f}}} (c_{\text{q}} u_{\text{q}}(t) + c_{\text{CO}_2} u_{\text{CO}_2}(s)) dt \right) \tag{8-6}$$

Thus, rewarding an increase of the EPI.

Three different initializations are used to learn for the EPI. First, learning the EPI from the randomized initialization. Second, the randomized$_{\mathrm{r}}$ initialization is used. Lastly, initialization 3 is used.

The simulations indicate that learning the EPI is a difficult task for the controller to achieve.

8-3-1 Results

The evolution of the EPI for all three different scenario's is shown in Figure 8-1. From these simulations, it is clear that learning for the EPI has not been done successfully, and a decrease in EPI is shown in all three cases. The relative decrease in EPI is different in all three scenario's.

Using the randomized initialization results in a decrease from 2.73 ($Hflm^{-2}$) to 2.72 ($Hflm^{-2}$), a decrease of less than half a percent of the EPI.

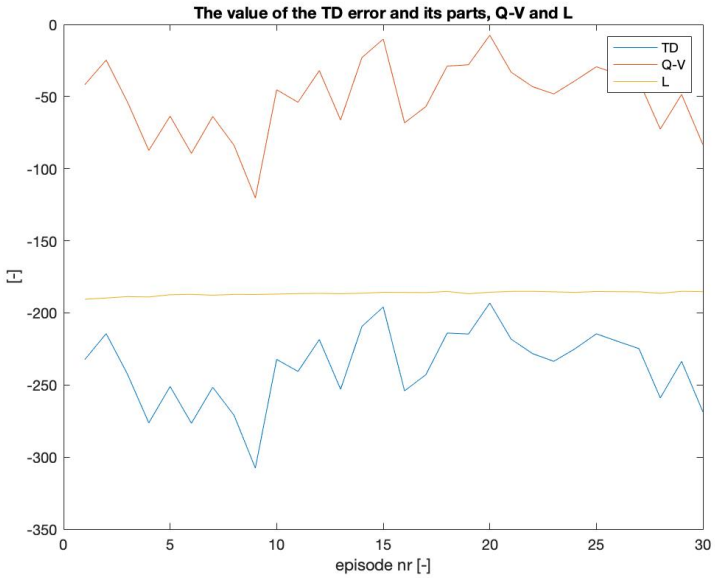

Figure 8-2: The evolution of the TD error during learning for the economic performance. Randomized initialization is used. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions.

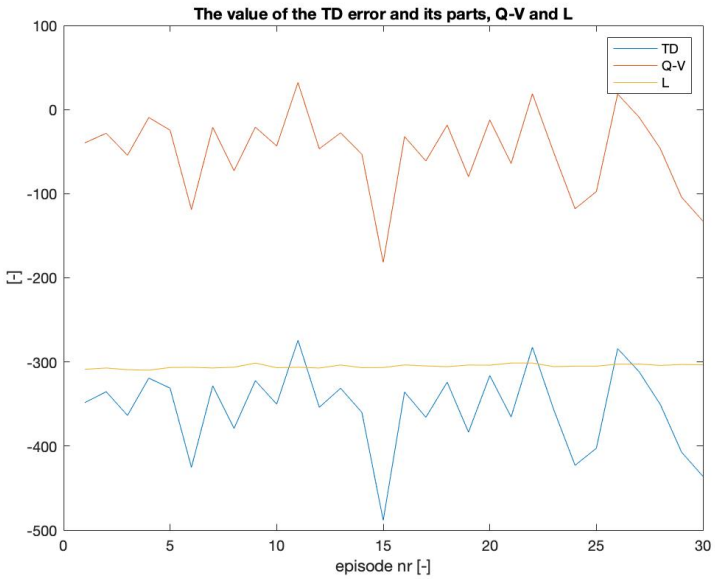

Figure 8-3: The evolution of the TD error during learning for the economic performance. Randomized$_{\mathrm{r}}$ initialization is used. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions.

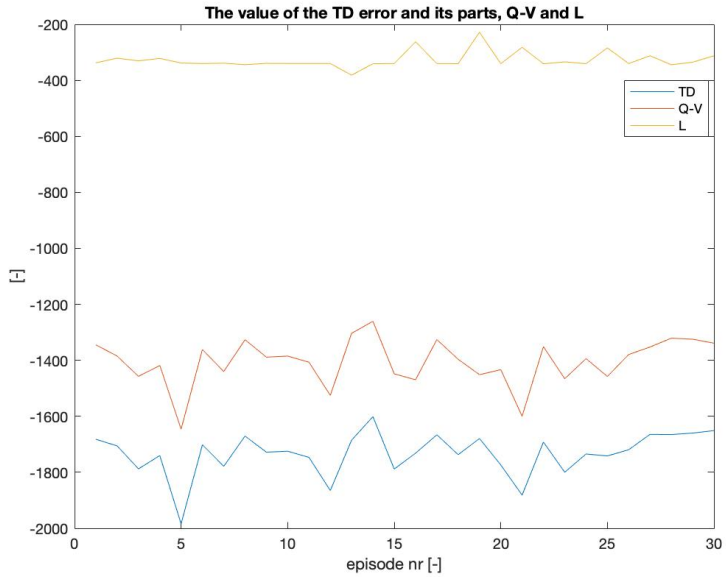

Figure 8-4: The evolution of the TD error during learning for the economic performance. The parameters are initialized as 1.1 times the nominal parameter value. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions.

Using the randomized$_r$ initialization, the EPI decreases by 1%. Interestingly, the constraint violations drop from 877 to 813, a decrease of 7.3%.

Initialization 3 shows no decrease in the EPI, but some significant spikes are present in the second half of learning. It is again noticed that the constraint violations decrease during learning, while this is not rewarded in the reward function.

In all three cases, the TD error, as shown in Figure 8-2, Figure 8-3, and Figure 8-4, does not show convergence to zero. Taking the reward function apart, plots of the evolution of yield and the sum of the heat inputs show that the yield is not increasing during learning and the heat input is increasing. This is noteworthy, since the RL reward function should punish increasing the heat input and reward the growth of the yield. Furthermore, it can be seen that the variations in the TD error from episode to episode are dominated by the difference in the value functions.

8-3-2 Conclusions

The simulations show that learning for the EPI with the RL reward function as shown in Equation 8-6 is not successful. All simulated cases show a decrease in the EPI with respect to the initial level. The behavior of the system during learning does not follow the pattern that can be expected with this reward function. It is surprising that no increase in EPI is possible whatsoever, therefor further research in the increase in one of its parts has been conducted. Some research into increasing the yield has been conducted in Appendix B. It is shown that solely learning for the yield is possible. The initialization of weight on the slack variables

an important factor in the learning of the yield. The learned model parameters also play an important role in learning the yield.

8-4 Case study 2: Decreasing constraint violations from perturbed initial parameters

The aim of this case study is to decrease constraint violations over time. In this case, performance based on EPI is not the main concern. The first interesting case is to initialize the system with the nominal model parameter values and try to find parameter updates that will decrease the constraint violations from there, ideally with only a minor decrease in EPI. The second case is to decrease constraint violations from the parameters that result in maximum EPI, these parameters cause quite significant constraint violations that could be harmful for the health of the crop.

In this case study the parameters are initializes in multiple ways. First, the randomized initialization is used. Secondly, initialization 1.1 is used. Lastly, simulations starting from the nominal model parameter values have been done, unfortunately some optimization error occurred that could not be solved within the time available.

The RL reward function is kept the same in both simulations:

$$\text{RL reward} = 0.04 * \sum_{k=t_\text{b}}^{t_\text{f}} (\max\{0, h_\theta(x_k, a_k)\}) \tag{8-7}$$

The factor for c_2, 0.04, is kept small such that the contribution of the constraint violations to the TD error is of the same order of magnitude as the difference between the value functions.

8-4-1 Results

In Figure 8-5 and Figure 8-6, the EPI and constraint violations of the system are shown while learning for minimizing constraint violations starting from the randomized initialization. The results indicate that a significant decrease in constraint violations is possible when starting from these initial parameter values. The best results are achieved when using initialization 1.1. The initial constraint violations are at a level over 504, while the final level is around 30, a decrease of 94% and well under the level achieved with the nominal parameter values.

Using the randomized initialization also results in a significant improvement of the constraint violations, but the improvement is not as good as with initialization 1.1. The initial constraint violation is at 297, and the final level is around 218, slightly above the level of constraint violations with the nominal parameter values. This decrease is about 26.6%.

The simulations show that the EPI decreases during learning. The EPI decreases from 2.73 to around 2.71 when the randomized initialization is used. When initialization 1.1 is used, the decrease is from 2.89, to around 2.3, a decrease of 20.5%.

The results achieved initialization 1.1 are notable because they are able to achieve constraint violations close to that of robust MPC, while still having economic performance almost 10% higher. The slack variables were expected to play the largest role in decreasing the constraint

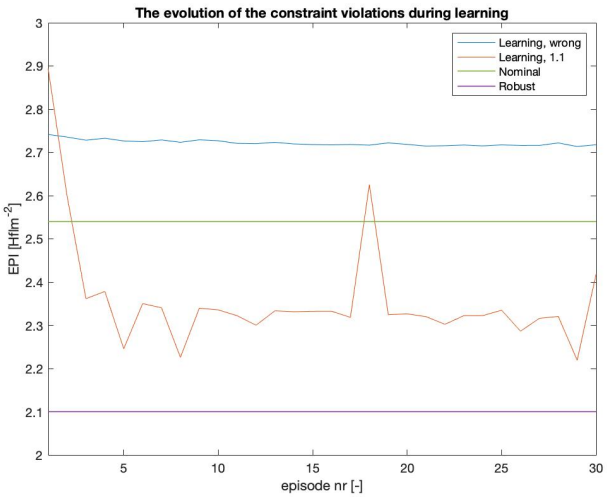

Figure 8-5: The evolution of the economic performance during learning for the constraint violations. In blue the evolution of the EPI is shown during learning, when the parameters are initialized as in Table 8-1. In orange the economic performance is shown when the parameters are initialized as 1.1 times the nominal value. In green the performance with nominal parameter values, and in purple the performance with samble-based MPC with 10 samples and 10% uncertainty. It can be seen that the economic performance stays close to its initial level in the first case, always being higher than in both the nominal and the robust case. In the second case, the performance drops significantly, but stays higher than in the robust case.

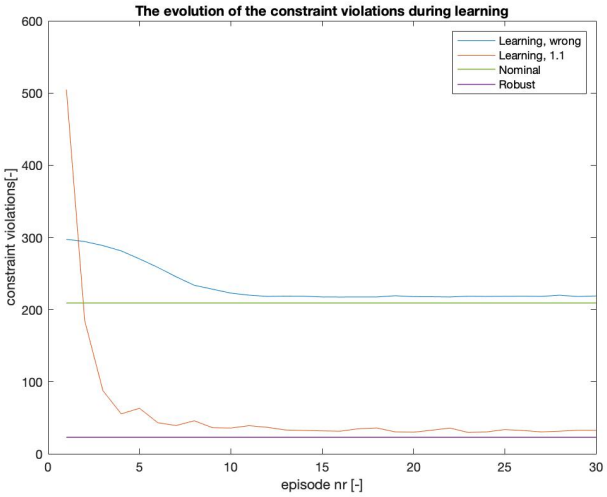

Figure 8-6: The evolution of the constraint violations during learning for the constraint violations. In blue the evolution of the constraint violations is shown during learning, when the parameters are initialized as in Table 8-1. In orange the evolution of constraint violations is shown when the parameters are initialized as 1.1 times the nominal value. In green the constraint violations with nominal parameter values, and in purple the constraint violations with samble-based MPC with 10 samples and 10% uncertainty. It can be seen that the constraint violations drop in both cases, but much more significantly in the second case.

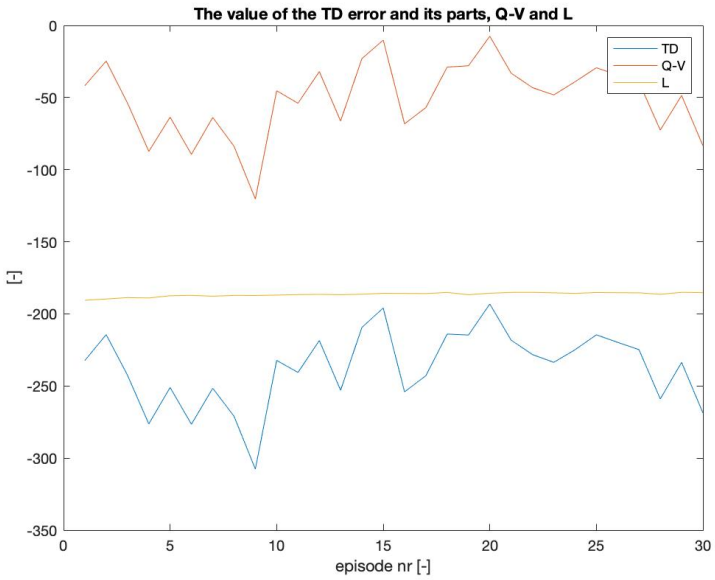

Figure 8-7: The evolution of the TD error during learning for the constraint violations. The parameters are initialized as in Table 8-1. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions. It can be seen that the reward function converges to around -185, while the difference between the value functions stays around the same level.

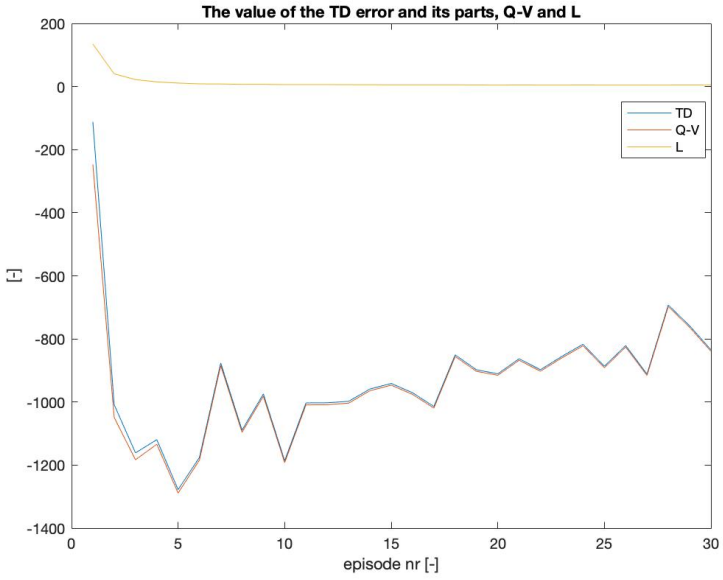

Figure 8-8: The evolution of the TD error during learning for the constraint violations. The parameters are initialized as 1.1 times the nominal parameter value. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions. It can be seen that the reward function converges to around 5, while the difference between the value functions first drops and then creeps up.

violations, by increasing the weight on the slack variables it is more costly to violate constraints. It is interesting to see that the weight on the slack variables is increased more in the case that the randomized initialization is used. In the randomized initialization case the weight increases to $3e4$, while in the initialization 1.1 case the weight is only increased to $2e4$.

Both cases do not show convergence of the TD error. This lack of convergence can mainly be attributed to the behavior of the value functions. The value of the TD error is dominated by the value functions.

Starting from the nominal parameter values has presented some unexpected issues. Due to unknown solver issues, the results stagnate after around 2000 time-steps and become unusable. Multiple scenarios were investigated, but unfortunately the issue could not be resolved in time.

8-4-2 Conclusions

It is clear that significant improvements of constraint violations are possible with the help of Q-ENMPC. It is shown that constraint violations can be decreased to a level below the level achieved with nominal parameters. The downside of decreasing the constraint violations is that the EPI also decreases. Comparing the results with those achieved with sample-based robust MPC shows that, when using initialization 1.1, the Q-ENMPC is able to achieve better economic performance while almost matching the constraint violations.

The fact that the EPI decreases when decreasing the constraint violations is expected. The main contributors to the constraint violations are the temperature in the greenhouse, which is often too low, and the humidity. The only way for the system to decrease both these factors is by increasing the heat input to the greenhouse. The heat input is one of the important factors that determine the EPI. Increasing the heat input will decrease the constraint violations, but also increase the cost to run the facility.

8-5 Case study 3: Improving EPI while decreasing constraint violations from perturbed initial modle parameter values

The final case study will focus on improving the EPI, while simultaneously decreasing constraint violations. The system will be initialized from the perturbed model parameter values. The goal of this case study is to continually improve the performance of the system, while not deteriorating its constraint satisfaction. A result that indicates performance close to a system operating with the nominal model parameter values and similar constraint satisfaction, while having started from the perturbed initial model parameter values, would already be a great achievement.

The values of c_1 and c_2 in the reward function, Equation 8-3, are decided to be -200 and 0.04 respectively. The learning rate is decreased at a rate of 10% per episode, allowing for learning even while the system is already mostly converged. The resulting RL reward function is as

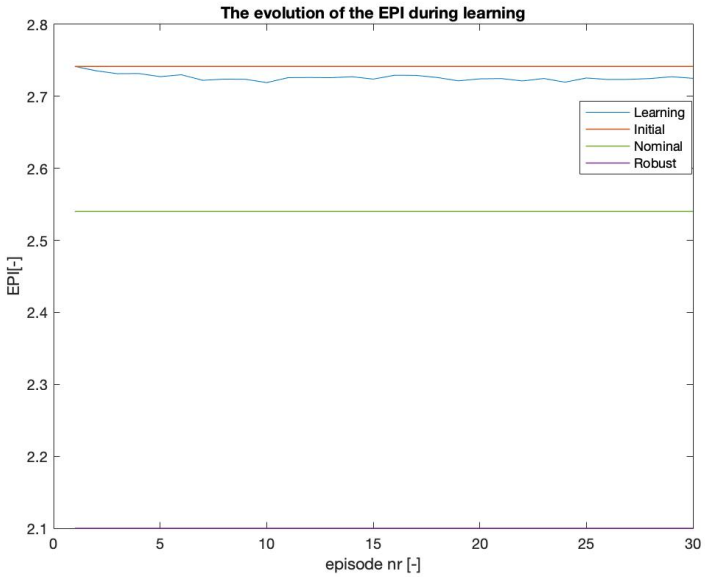

Figure 8-9: The evolution of the economic performance during learning for both the economic performance and the constraint violations. The parameters are initialized as in Table 8-1. In blue the evolution of the EPI is shown during learning, in orange the initial performance, in green the performance with nominal parameter values, and in purple the performance with samble-based MPC with 10 samples and 10% uncertainty. It can be seen that the economic performance stays close to its initial level, albeit slightly lower.

follows:

$$\begin{aligned} \text{RL reward} = -200 * \Big(c_{\text{pri,s}} y_1(t_\text{f}) - \sum_{k=t_\text{b}}^{t_\text{f}} (c_\text{q} u_\text{q}(t) + c_{\text{CO}_2} u_{\text{CO}_2}(s)) dt \Big) \\ +0.04 * \sum_{k=t_\text{b}}^{t_\text{f}} (\max\{0, h_\theta(x_k, a_k)\}) \end{aligned} \tag{8-8}$$

8-5-1 Results

In Figure 8-9, Figure 8-10, and Figure 8-11 the EPI, constraint violations and TD error of the system are shown, respectively, while learning for performance and minimizing constraint violations using randomized initialization. The Q-ENMPC scheme is able to reduce the constraint violations by 30%, while keeping the performance around the starting level. In this case the starting level is already a performance 7.5% higher than the performance with nominal parameters. After learning, the constraint violations are only 4.8% higher than in the nominal case.

The TD error decreases from 297, to around 219, a decrease of 26.3% with respect to the starting level. While this is better than the starting point, convergence by means of the TD error is not shown. The remaining value of the TD error is a result of the difference between the value functions, as can be seen in Figure 8-11. The convergence that is seen is mainly a

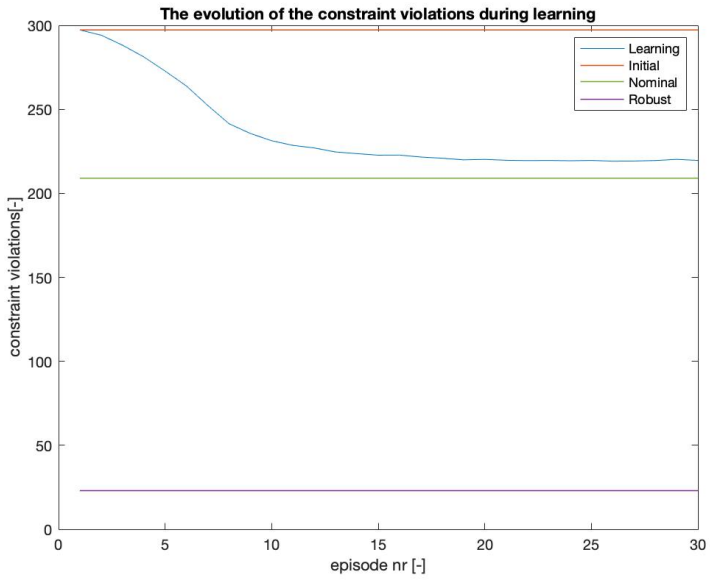

Figure 8-10: The evolution of the constraint violations during learning for both the economic performance and the constraint violations. The parameters are initialized as in Table 8-1. In blue the evolution of the EPI is shown during learning, in orange the initial performance, in green the performance with nominal parameter values, and in purple the performance with samble-based MPC with 10 samples and 10% uncertainty. It can be seen that the constraint violations decrease significantly during learning. The final performance is similar to that with nominal parameter values, but much higher than in the case of robust MPC.

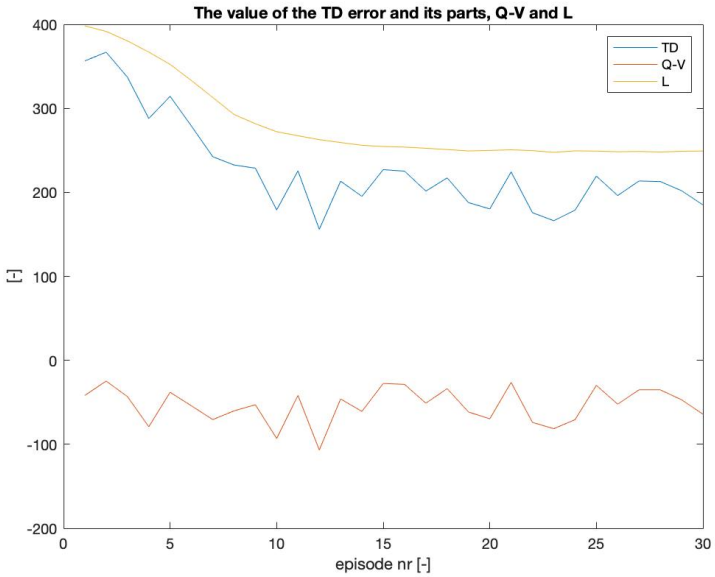

Figure 8-11: The evolution of the TD error during learning for both the economic performance and the constraint violations. The parameters are initialized as in Table 8-1. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions. It can be seen that the reward function converges to around 250, while the difference between the value functions stays around the same level.

result of the decrease of the learning rate. The value of the TD error is dominated by the difference between the value functions.

Constraint violations are only seen in the temperature and humidity constraints. The temperature bounds are slightly violated both on the high and low ends, this is deemed too little to be of concern. The humidity is constantly being violated on the high end. It is quite logical to violate this bound, as a high humidity helps with crop growth according to the model. The bounds are in place to prevent crop disease, which is not explicitly accounted for in the model. Literature reports higher chances of crop disease at elevated humidity levels, however, studies are conducted at humidity levels of 80% or higher, which is significantly higher than the average learned humidity level of 71%. It can also be seen that at the start of the learning process the average humidity is at over 73%, which is significantly higher than the average humidity at the end of learning.

8-5-2 Conclusions

In this case study the goal was to improve both the performance with respect to the EPI and with respect to the constraint violations. Improving the performance with respect to the constraint violations has been shown, but the EPI has decreased. When comparing the results from this case study with those of the second case study, where only decreasing the constraint violations was the goal, the results are very similar. In this case study the decrease in constraint violations is slightly less, and the decrease in EPI is also slightly better, but the

difference is barely noticeable.

8-6 Results and Conclusions

In this chapter simulations were performed with the Q-ENMPC approach for various case studies. The case studies aimed at decreasing constraint violations, increasing performance in terms of EPI and a combination of both.

From these case studies it is evident that it is possible to significantly decrease constraint violations starting from the perturbed parameter values. Increasing the EPI has proven difficult in all cases.

In the first case study, it is shown that learning for the EPI with the RL reward function as shown in Equation 8-6 is not successful. All simulated cases show a decrease in the EPI with respect to the initial level. The behavior of the system during learning does not follow the pattern that can be expected with this reward function. It is shown that solely learning for the yield is possible. The initial weight placed on the slack variables is an important factor in the learning of the yield.

In the second case study it is shown that significant improvements of constraint violations are possible with the help of Q-ENMPC. It is shown that constraint violations can be decreased to a level below the level achieved with nominal parameters and almost equal to the level achieved with robust MPC in some cases. As expected, the downside of decreasing the constraint violations is that the EPI also decreases.

Lastly, the goal was to improve both the performance with respect to the EPI and with respect to the constraint violations. Improving the performance with respect to the constraint violations has been shown, but the EPI has decreased. When comparing the results from this case study with those of the second case study, where only decreasing the constraint violations was the goal, the results are very similar. In this case study the decrease in constraint violations is slightly less, and the decrease in EPI is also slightly better, but the difference is barely noticeable.

With the tested reward functions, parameter value initializations, and updated parameters it is seen that a decrease of the constraint violations is possible, while, with the tested reward functions, an increase in the EPI was not possible. To increase the yield, which has been possible on its own, it was beneficial to place no weight on the slack variables. The most notable result from the case studies is that it is possible to achieve constraint violations on a similar level as with robust MPC, while having a higher economic performance.

Chapter 9

Conclusions and discussion

In this thesis, data driven ENMPC using Q-learning is proposed to increase the performance of a greenhouse growing lettuce. Various case studies are performed and results are compared to those of existing research. In this chapter, research questions are answered, final conclusions are drawn and discussed, and recommendations for future research are provided.

9-1 Conclusions

The main research questions of this work was:

Can ENMPC using reinforcement learning be used in greenhouse climate control and improve performance compared to robust sample-based MPC in the presence of parametric uncertainty?

The sub-questions that were investigated in the case studies are:

- *Can ENMPC using RL increase economic performance, starting from perturbed initial model parameter values?*
  This sub-question can be answered with a definite no. The reward functions and initializations tested in this thesis have not resulted in an increase of the EPI.

- *Can ENMPC using RL decrease constraint violations, starting from the nominal model parameter values?*
  ENMPC using RL can decrease constraint violations significantly. In the case studies it is shown that the constraint violations can be decreased by 30% when initializing the system with perturbed parameter values, even reaching a decrease of over 90% in certain cases. Unfortunately, starting from the nominal model parameter values has resulted in solver issues that could not be solved in time.

- *Can ENMPC using RL increase economic performance and decrease constraint violations at the same time, while starting from perturbed initial model parameter values.* The answer to this sub-question is that Q-ENMPC has not shown to increase the economic performance, while decreasing the constraint violations. No cases have resulted in an increase of the EPI, while decreasing the constraint violations was possible in many cases.

Because all sub-questions are answered, it is now possible to answer the main research question. In this thesis, performance was defined both with an economic indicator and based on constraint violations. The simulations-based case studies in this thesis show that an increase of the performance, based on constraint violations, is possible. Significant decreases in constraint violations have been shown in the case studies, suggesting that Q-ENMPC is well able to increase performance in that regard. During the case studies, the constraint violations have been decreased by up to 94%. Improving the performance based on EPI has been proven difficult. The case studies conducted in this thesis have resulted in an improvement of the EPI compared to the level achieved with robust MPC, while decreasing constraint violations to a similar level.

A data-driven ENMPC approach using Q-learning was proposed in this thesis to overcome the challenges posed by parametric uncertainty in greenhouse climate control. Three case studies were designed to test the approach on a greenhouse climate model with a lettuce crop. Performance was tested based on the EPI and constraint violations. In those case studies it has become clear that significant performance increases can be achieved in terms of constraint violations, but increasing the EPI is not possible with the tested reward functions. The overall performance, when compared to that of robust MPC is increased by almost 10%. A draft of a paper based on these findings is attached to this thesis in Appendix C

9-2 Discussion

The implications of this work are worthy of discussion. From the results it is evident that the economic performance and the constraint adherence of a greenhouse growing lettuce can be improved significantly. Relevant questions should be asked about the validity and applicability of these results.

First, it is not clear what performance is currently achieved in commercial greenhouses. The results of this work are compared to previous academic results, which indicate upside potential. Commercial, rule-based, controllers have not been checked for their EPI or constraint violations.

Secondly, a single episode encompasses one growth cycle, or 40 days in the real world. The results indicate convergence after well over 15 episodes which is over 1.5 years of real world time. Most farmers are already hesitant of implementing control techniques in their farm that are unknown to them, having to wait for over a year before optimal performance is reached might discourage further. It should be noted that performance at the start is not necessarily bad, but the constraint violations could prove prohibitive.

Thirdly, the value for the constraints is not set in stone. Currently, the constraint boundaries are decided upon based on rules of thumb in the field, which is based on research. From the

simulations it is clear that much higher economic performance is theoretically possible, as long as the lettuce doesn't get any disease. The specific conditions for plant disease are still a topic of research, and much research around high humidity focuses on humidity levels of over 80%, while levels of 75% already provide greatly improved economic performance.

Fourth, starting from different initializations results in significantly different results. The source of these variations is most likely the non-linearity in the model. The model is highly nonlinear with many local minima. It is beyond the scope of this work to work out the global optimum, but is important to be aware of this aspect of the model used. There are many strategies to overcome local minima and work towards a global optimum, this could be part of future research based on this thesis.

Fifth, in Appendix B it is shown that the parameters that are updated after each episode have a significant influence on the performance of the controller. In the case studies in chapter 8, the same parameters are updated in all case studies. Since the chosen parameters have such an influence on the resulting performance, a new parameter selection should be conducted for every case study.

Computation time was a major limiting factor in this study, simulation of a single episode took from around 45 minutes up to multiple hours, resulting in a single simulation taking almost a day, but often much longer, to simulate 30 episodes. Since nothing is perfect the first (or the second... or third...) time around, reaching satisfying results for a single case study could take multiple weeks. The amount of episodes during learning was decided to be 30. In combination with the decreasing learning rate, this resulted in a meaningful amount of updates without simulations taking too long. Still, some simulations could take over 3 days to finish. Unfortunately, the amount of episodes meant that convergence, in terms of the TD error going to zero, was never reached in any of the case studies. Longer simulations with more episodes have been attempted, but convergence of the TD error was never really shown. Some simulations have come close, but steady convergence to zero has never been the case. One important factor in this result is the decreasing learning rate, by decreasing the learning rate the update would become too small at some point to deliver significant changes in the TD error. Longer simulations without a decrease in learning rate have resulted in solver issues at some point during the simulation, never actually reaching convergence of the TD error.

9-3 Future work

This research has already shown some promising results. Unfortunately, not everything can be investigated in one thesis study. Some suggestions for future work are given here:

Further analysis in parameter choice

All case studies in this work update the same set of model and ENMPC parameters. It is shown in Appendix B that the choice of model parameters to update has a significant impact on the learning behaviour and final performance of the controller. In this work some foundations have been laid for determining which parameters should be updated in certain cases, in future work, rigorous research should be done into the specific parameters that are best updated for each case study.

Multiple parameter updates per episode

Over the course of an episode the gradient of the system changes significantly. Both over the course of a day, influenced by the day and night rhythm and radiance, as well as over multiple days, influenced by the growth of the lettuce. The chosen method of dealing with this in this work was to only apply one parameter update per episode and use the mean gradient during the episode for that update. Finding a method to deal with the changing gradient during the episode and applying multiple updates potentially allows to capture more information during the episode.

Convergence

In this work, simulation time was a prohibitive factor to the results. Measures were taken to make the program less computationally expensive and run the program simultaneously on multiple devices. It was quickly found that, especially during development, running multiple case studies at once would not speed up the initial stages of research. Results were needed to build upon and solve issues. Running into the same issue on multiple devices at the same time would not increase the research speed. Considering the length of the simulations, an easy step for future work would be to run the various case studies towards convergence of all aspects, as in this work time limitations prohibited that.

LSTDQ-Learning

The simulations have shown the potential of using ENMPC as a function approximator of RL. In this work a simple Q-learning RL approach was taken. In theory, more information can be included in the update steps when taking into account the Hessian information of the system. One method of including this information is by using an LSTDQ-learning method, described in [57]. One of the potential benefits is that by including the Hessian information of the system, one can in theory make much larger update steps and reach convergence in less steps. Reaching convergence in less steps would make this technology would appealing.

Time-dependent parameters

Potentially, certain parameters can be identified that aid in the growth of the crop. It could be the case that in the early growth stages the temperature is much more important than at the end of the crop cycle, while the CO_2 level is much more important at the end of the growth cycle. This information could by identified in future work and a method can be implemented to include this information.

Comparison to reality

Unfortunately, a clear comparison to reality is not possible in this work. Comparisons have been made to existing research, but comparisons to the commercial field would be relevant and interesting for real world implementation. Future work could focus on bridging this gap, finding real world examples of the amount of constraint violations and the EPI. Using

the same weather scenario during simulations could then help determine if improvement is possible.

New plant models

This work uses the greenhouse-lettuce model by van Henten. The theory discussed in this research could be used for any other crop grown in a greenhouse. The reasons for using a lettuce model have been discussed in this work, and extra work will be necessary to take other crops into account. The application of this theory to other crops is not straightforward, but interesting nonetheless.

Bridging the gap between theory and reality

Once the theory discussed in this work has been thoroughly proven, it should be implemented in a real wold greenhouse. The first time this is done it should be in a environment where all greenhouse aspects are well-known beforehand or even one where the weather aspects can be controlled. The benefit of this highly controlled environment is that the step from simulation to reality is smaller, as the main difference will be the plant itself and not the outside world. Once the results of that experiment are satisfying, experiments in a commercial greenhouse can be done.

Appendix A

Exploration

During simulations it was discovered that exploration has a significant influence on the learning process. Initially this influence is positive, but the overall learning performance seems to deteriorate by the exploration.

The case below is equal to case 3 discussed in chapter 8, the parameter value initialization is the same and the reward function is the same. The only difference is that exploration is included in these cases, starting at a value of 5000 and decreasing every episode (the value of the exploration parameter is multiplied by 0.613 after every episode). Here the influence of exploration on the learning behavior can be seen.

In the study, when including exploration, a good value for the constraint violations is found in episode 5. The influence of keeping the decay rate of the exploration constant, keeping the value for the exploration parameter constant, and setting the value for the exploration parameter to zero after 5 episodes is compared to the learning behavior of not having any exploration and the randomized initialization.

In all cases where exploration was initially present, compared in Figure A-1 and Figure A-2, it can be seen that the constraint violations and EPI converge to the same level. This is the performance level achieved with the randomized initialization and no exploration in the first episode. No performance increase is seen whatsoever when comparing the final performance of the cases where exploration was present and decayed to zero, or the initial performance when exploration is not included.

When exploration is kept constant after 5 episodes, the constraint violations and EPI also stay constant. Again, no learning behavior on the model parameter values is seen.

In the case that exploration is not included, learning behavior of the parameter values is significant. It is thus concluded that including exploration in this research is not beneficial to the learning behavior and exploration will not be considered in this thesis work.

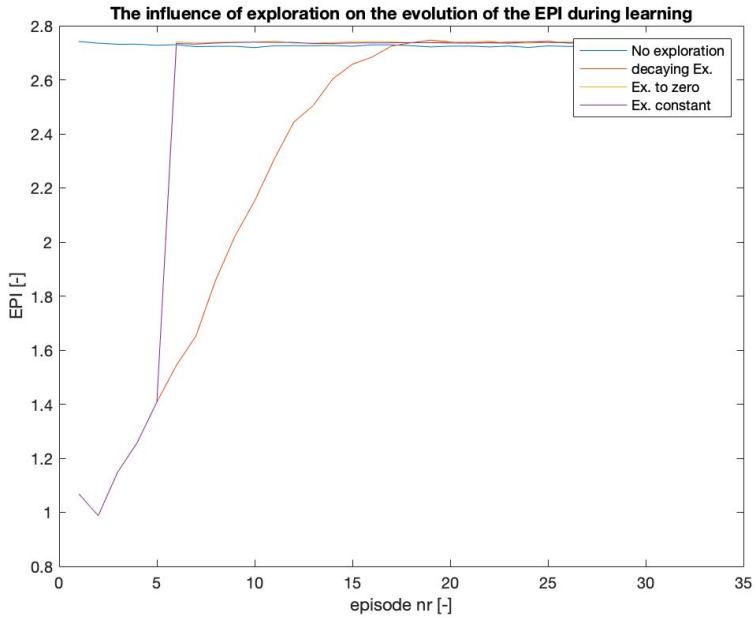

Figure A-1: The influence of the exploration on the evolution of the EPI during learning. No exploration is present in blue, in orange a constantly decaying value for the exploration parameter is present, in yellow the value for the exploration parameter is set to zero after 5 episodes, in purple the value for the exploration parameter is kept constant after 5 episodes.

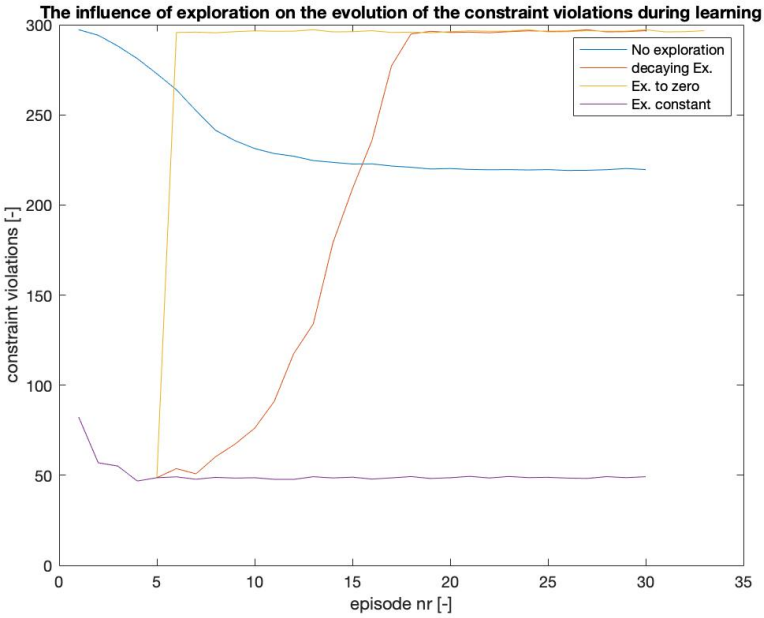

Figure A-2: The influence of the exploration on the evolution of the Constraint violations during learning. No exploration is present in blue, in orange a constantly decaying value for the exploration parameter is present, in yellow the value for the exploration parameter is set to zero after 5 episodes, in purple the value for the exploration parameter is kept constant after 5 episodes.

Appendix B

Increasing the yield

In this chapter increasing the yield starting from perturbed initial model parameter values in investigated. This research was not initially part of this thesis work and thus comprises simulations that are not as coherent as the case studies discussed in chapter 8.

It was seen that learning for the yield can be done quite successfully, The reward function when learning for the yield alone is much simpler:

$$RLreward = -10 * (y_1(t_f) \tag{B-1}$$

Multiple simulations are conducted and they are initialized in various manners.

B-1 Results

Results of learning for the yield are shown in Figure B-1, Figure B-2, and Figure B-3. In these plots it is seen that the yield increases in all instances, but significantly more in the case that the weight on the slack variables is put to zero and different parameters are updated than in all other case studies. In Figure B-3, the yield factor, light use efficiency, effective canopy surface, and to a lesser extend the temperature influence on the photosynthesis are updated after each episode. In Figure B-1 and Figure B-2, and all other case studies, the CO_2 capacity of the greenhouse, heat capacity of the greenhouse air and coefficient of leaf-air vapor flow, for the CO_2 concentration, temperature and humidity are updated after each episode. In the first case, all parameters that are updated are determined to have the largest possible influence on the yield, while in the second case the influence on the inputs is also considered.

An important consideration in these cases is that the learning rate is initialized such that the weight on the slack variable changes by 50% after the first episode. If the weight is initially $5e3$ this will change to $7.5e3$ after the first episode, while if the weight is initially zero it will not change. The yield increase is 1.7% when the weights on the slack variables are initialized at $5e3$, against 28.7% when they are zero and different parameters are updated.

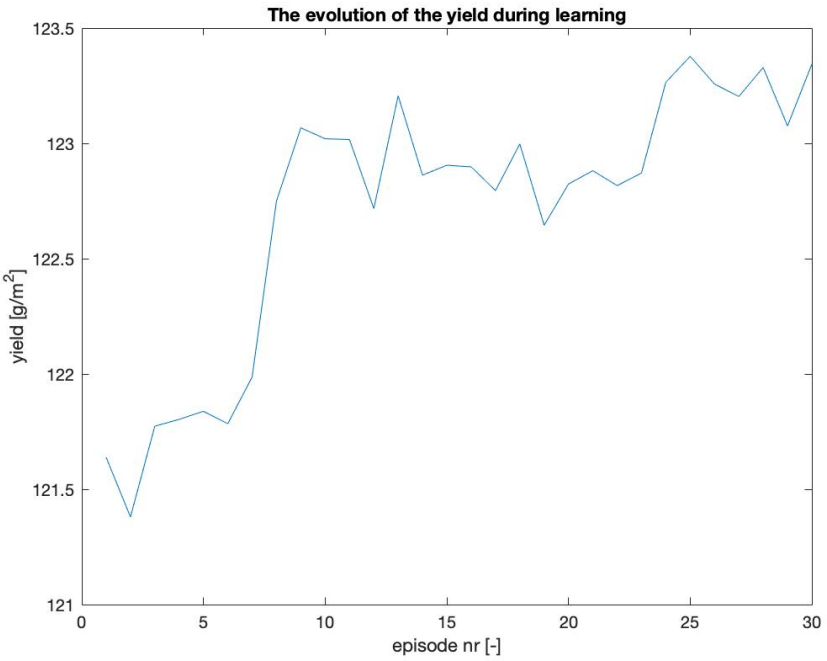

Figure B-1: The evolution of the yield during learning for the yield alone. The randomized initialization is used for the model parameters. In blue the evolution of the yield is shown during learning. It can be seen that the yield increases during learning.

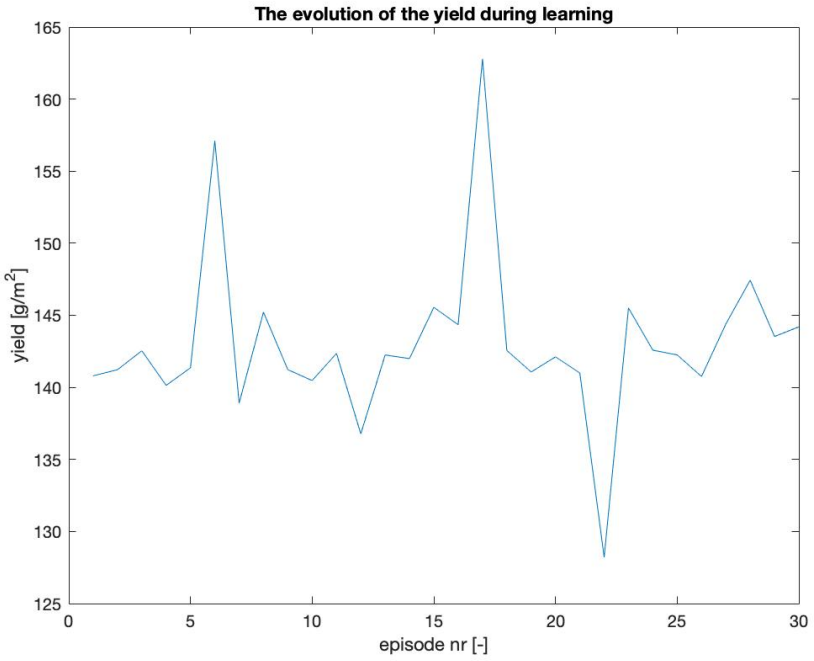

Figure B-2: The evolution of the yield during learning for the yield alone. Randomized initialization is used, with no weight on the slack variables. In blue the evolution of the yield is shown during learning. It can be seen that the yield increases during learning.

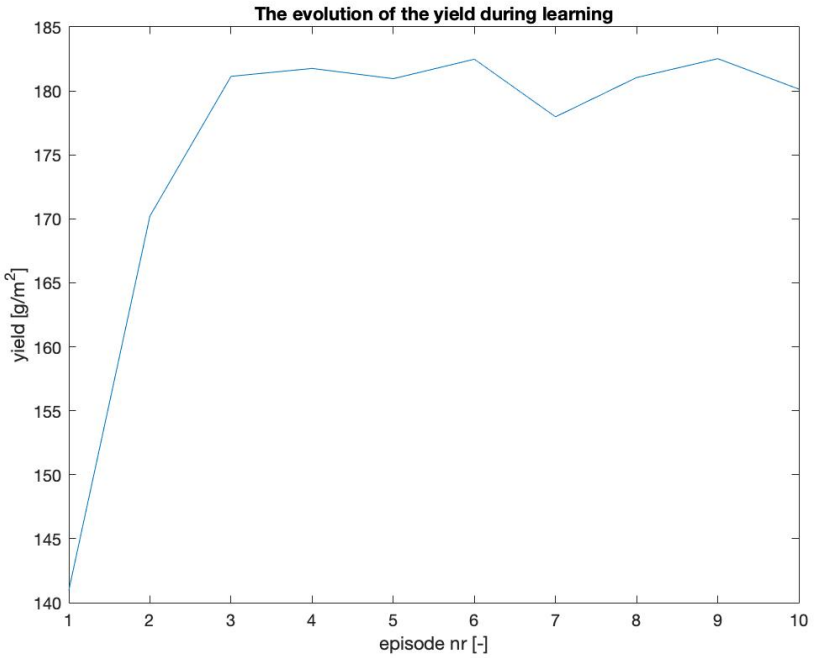

Figure B-3: The evolution of the yield during learning for the yield alone. Randomized initialization is used, with no weight on the slack variables. Furthermore, different parameters are updated and the learning rate is not decreased after each episode. In blue the evolution of the yield is shown during learning. It can be seen that the yield increases during learning.

Appendix C

Paper draft

A draft version of a paper describing the findings of this thesis is attached. The paper focuses on increasing the performance of the system based on the constraint violations.

Greenhouse climate control with data-driven ENMPC using reinforcement learning

Seymour Lubbers, Name Surname 2,

***Abstract*—Greenhouses allow production of crops that would otherwise be impossible. Permitting more local, fresher and nutrient richer crop production. Efforts are taken to minimize societal harm due to energy and resource consumption by greenhouse production systems. One way to control such systems is by using model predictive control. Optimal crop yield and resource efficiency can, in theory, be achieved by model predictive control. Unfortunately, one major drawback of model predictive control is that it is not well equipped to deal with uncertainty. Significant prediction errors can occur when a mismatch between the model and the real system exists, resulting in deteriorated performance of the system. Strategies exist, such as robust MPC, that are designed to handle uncertainty, but those often result in conservative control policies. This thesis proposes to use model predictive control as a function approximator for RL in order to learn model and MPC parameters that can deliver optimal performance in the case of model mismatch.**

In this paper, data-driven economic nonlinear model predictive control using reinforcement learning is proposed as a method to alter the model parameters. The performance of the system after learning is compared to approaches using robust and nominal model predictive control. The goal in this paper is to decrease constraint violations, starting from perturbed initial model parameter values. The simulation results show that the data-driven ENMPC using reinforcement learning is able to achieve economic performance 10% higher than when the system is controlled using robust control, while constraint violations are similar. The constraint violations are decreased by more than 90% during the learning process.

***Index Terms*—Model predictive control (MPC), reinforcement learning (RL), greenhouse climate control, function approximation, adaptive nonlinear model predictive control, economic NMPC (ENMPC).**

I. Introduction

Greenhouses shelter crops, enable control over crop production, allow to boost production, permit the control of quality, have production of crops that would otherwise be impossible, and prolonging of the cultivation period [1]. Energy is needed for the climate control of the greenhouse, and water and nutrients, including CO_2 must be supplied from the outside. Societal demands for fresh, safe, and price-worthy foods [2], in combination with sustainability concerns make it necessary to optimize greenhouse operations. A consequence of this combination of demands is that one can't simply increase inputs to grow better produce, this would dramatically lower the number of crops grown per unit of energy or nutrient input.

Seymour received the B.Sc degree in Mechanical Engineering from Delft University of Technology, Delft, The Netherlands in 2019. He is currently finishing his M.Sc degree in Systems and Control from Delft University of Technology, Delft, The Netherlands, focusing on using data-driven ENMPC using reinforcement learning in greenhouse climate control applications. E-mail address: {s.z.lubbers@student.tudelft.nl.

Energy usually accounts for the second highest overhead cost in greenhouse crop production [3]. In the Netherlands, the cultivation of fruits and vegetables in greenhouses requires an enormous amount of energy. 77% of the total $100.5\ PJ$ was due to heating and 23% due to electricity for supplemental lighting [4]. The Dutch horticultural industry has agreed with the Dutch government to decrease the total energy consumption [4].

Optimal control, and its derivative MPC, have grabbed the attention of researchers for many years already [5]. Because this control method allows capturing system knowledge and, therefore, offers an opportunity to capture plant dynamics to a higher degree than growers themselves can [2]. A second opportunity for optimal control is offered by reinforcement learning. This machine learning approach can learn system dynamics by trial and error, also capturing system uncertainties in the process. Being able to deal with system uncertainties is a welcome characteristic of reinforcement learning that standard MPC approaches do not offer. However, reinforcement learning also comes with its shortcomings, for one learning from trial and error dismisses any prior knowledge about the system and its dynamics.

Both approaches offer advantages that neatly overcome the disadvantages of the other strategy. Research towards combining the strengths of MPC and RL is rising and some promising results are achieved [6] [7] [8]. Nonetheless, little research can be found on combining MPC and RL in the field of greenhouse climate control up to this date.

The objective of this study is to improve the performance of greenhouse climate control in uncertain conditions, by using MPC as a function approximator for RL. Since model parameters can differ great from greenhouse to greenhouse and in most cases are not known with great certainty, MPC can struggle with providing consistently good performance. This study aims to overcome the problems that parametric uncertainty poses when MPC is used to control the greenhouse climate, while maintaining the ability to incorporate system knowledge in the controller. This is done by using MPC as a function approximator in a Q-learning approach.

This paper is organised as follows: First the greenhouse climate and crop model is discussed in section II. Then, the Q-ENMPC approach is discussed in section III, with the set-up for the case study presented in section IV. Numerical simulations are ran and the results are presented in section V. Conclusions are drawn in section VI.

II. Greenhouse lettuce model

The model that is used in this work is the nonlinear model first presented in [9], previously used in [10] and [11]. The model is discretized using the fourth-order Runge Kutta method. This results in

$$x(k+1) = f(x(k),\ u(k),\ d(k), p), \tag{1}$$
$$y(k) = g(x(k),\ p) \tag{2}$$

with discrete time $k \in \mathbb{Z}^{0+}$, state $x(k) \in \mathbb{R}^4$, input $u(k) \in \mathbb{R}^3$, disturbance $d(k) \in \mathbb{R}^4$and $f(\cdot),\ g(\cdot)$ nonlinear functions.

The state, input and disturbance signals are further defined as:

$$x(k) = \begin{pmatrix} W(k) & C_{CO_2,a}(k) & T_a(k) & C_{H_2O,a}(k) \end{pmatrix}^T \tag{3}$$
$$= \begin{pmatrix} x_1(k) & x_2(k) & x_3(k) & x_4(k) \end{pmatrix}^T \tag{4}$$
$$u(k) = \begin{pmatrix} u_{CO_2,o}(k) & u_v(k) & u_q(k) \end{pmatrix}^T \tag{5}$$
$$= \begin{pmatrix} u_1(k) & u_2(k) & u_3(k) \end{pmatrix}^T \tag{6}$$
$$d(k) = \begin{pmatrix} I(k) & C_{CO_2,o}(k) & T_o & C_{H_2O,o}(k) \end{pmatrix}^T \tag{7}$$
$$= \begin{pmatrix} d_1(k) & d_2(k) & d_3(k) & d_4(k) \end{pmatrix}^T \tag{8}$$

The state $x(k)$ contains the dry matter content of the lettuce, W, in kg$\cdot$m^{-2}, which is the lettuce's weight per square meter, after all its water has been removed. The crop dry matter is often used because it shows less seasonal variation than fresh weight. Furthermore, the state contains the indoor carbon dioxide concentration, $C_{CO_2,a}$, in kg $\cdot$ m^{-3}, air temperature, T_a, in °C, and humidity, $C_{H_2O,a}$, in kg $\cdot$ m^{-3}.

The control input $u(k)$ contains the rate of carbon dioxide supply, u_{CO_2}, in mg $\cdot$ m^{-2} $\cdot$ s^{-1}, ventilation rate through the window vents, u_v, in mm $\cdot$ s^{-1}, and energy supply by the heaters, u_q, in W $\cdot$ m^{-2}.

The weather disturbance, $d(k)$, contains the radiation from the sun, I_o, in W $\cdot$ m^{-2}, the outside carbon dioxide concentration, $C_{CO_2,o}$, in kg $\cdot$ m^{-3}, outdoor temperature T_o, in °C, and humidity, $C_{H_2O,o}$, in kg $\cdot$ m^{-3}.

The measurements, $y(k) \in \mathbb{R}^4$, contain the dry weight matter, W, in g$\cdot$m^{-2}, the indoor carbon dioxide concentration, $C_{CO_2,a}$, in ppm, the indoor air temperature, T_a, in °C, and relative humidity, $C_{H_2O,a}$, in %. It should be clear that $y(k)$ contains equivalent states as $x(k)$, but with different units for the dry weight, indoor carbon dioxide concentration and humidity.

In Figure 1, the greenhouse model with lettuce is graphically depicted. The interaction between the states, control inputs, and disturbances is indicated by arrows. From this figure it is clear that the control inputs and weather disturbances influence the greenhouse climate, which in turn influences the lettuce. The only exception to this is the solar radiation, which also has a direct influence on the crop. All three control inputs have a direct influence on the greenhouse climate, while having an indirect influence on the lettuce's dry weight. It is via controlling the greenhouse climate that the lettuce growth is controlled.

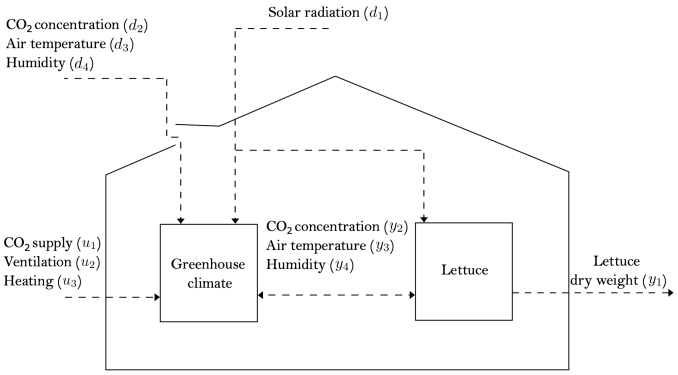

Fig. 1: Greenhouse with lettuce schematically represented. The interaction between the greenhouse and lettuce model is indicated with the arrows. The influence of the control signal $u(k)$ and weather disturbance $d(k)$ are also indicated by the arrows. The figure is taken from [11].

A. Uncertainty in this work

The kind of uncertainty that will be considered in this work is parametric uncertainty, equal to the uncertainty considered in [10]. In this work, it is assumed that the model parameters can be modeled as an uncertain parameter with known probabilistic properties. It is assumed that:

$$\hat{p} \sim \mathcal{U}\Big(\mu_{\hat{\text{p}}}, \Sigma_{\hat{\text{p}}}(k)\Big), \tag{9}$$

with uniform distribution $\mathcal{U}(\cdot)$ having a mean value of $\mu_p = p$ and co-variance matrix $\Sigma_{\hat{p}} = \mathbb{E}\big[(\hat{p} - \mu_{\hat{p}})(\hat{p} - \mu_{\hat{p}})^T\big]$. with expectation operator $\mathbb{E}$. Furthermore, it is assumed that $\Sigma_{\hat{p}}$ is a diagonal matrix, meaning that there is no cross-covariance. The justification for a uniform distribution is two-folded. Firstly, certain parameters can not be negative or outside certain ranges. Secondly, it is undesired to have an infinite support (such as a Gaussian distribution). Due to the assumption of having an uncertain parameter in the measurement model, the state and measurement estimation are also uncertain. The system model is then depicted as:

$$x(k+1) = f(\hat{x}(k),\ u(k),\ d(k), \hat{p}), \tag{10}$$
$$y(k) = g(\hat{x}(k),\ \hat{p}) \tag{11}$$

where $\hat{p}$ indicates the uncertain parameters.

B. Optimization problem

The goal is to achieve maximum dry matter production, with the least amount of energy input as possible. It is assumed that at each time instant, the state $x(k)$ can be measured. Then, the following optimization problem is solved at each time step:

$$
\begin{aligned}
\min_{u(k)} \; & \sum_{i=1}^{Ns} \sum_{k=k_0}^{k_0+N_P} V\big(u(k), \hat{y}^i(k)\big), \\
\text{s.t. } & \text{Equation 1}, \; \hat{x}(k_0) = \hat{x}_0, \\
& u_{\min} \leq u(k) \leq u_{\max}, \\
& |u(k) - u(k-1)| \leq \delta u, \\
& \hat{y}_{\min}(k) \leq \hat{y}^i(k) \leq \hat{y}_{\max}(k), \text{for } i = 1, \ldots, N_s, \\
& \text{and for } k = k_0, \ldots, k_0 + N_P,
\end{aligned}
\tag{12}
$$

with $N_P \in \mathbb{R}$ the prediction horizon. The value $N_s \in \mathbb{R}$ represents the number of realisations taken from Equation 9. In this work the number of realisations taken from the distribution will be 1. The comparison work utilizes sample-based robust control and takes multiple realisations.

The constraints in Equation 12 are defined:

$$u_{\min} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T, \tag{13}$$

$$u_{\max} = \begin{pmatrix} 1.2 & 7.5 & 150 \end{pmatrix}^T, \tag{14}$$

$$\delta u = \frac{1}{10} u_{\max}, \tag{15}$$

$$\hat{y}_{\min} = \begin{pmatrix} 0 & 0 & f_{\hat{y}_{3,\min}}(k) & 0 \end{pmatrix}^T, \tag{16}$$

$$\hat{y}_{\max} = \begin{pmatrix} \inf & 1.6 & f_{\hat{y}_{3,\max}}(k) & 70 \end{pmatrix}^T, \tag{17}$$

with lower and upper bound on the control signal defined by $u_{\min}$ and $u_{\max} \in \mathbb{R}^3$, and the bound on the change of the control signal defined by $\delta u \in \mathbb{R}^3$. These bounds correspond to current horticultural practice. The lower and upper bound on the output are $\hat{y}_{3,\min}(k)$ and $\hat{y}_{3,\min}(k) \in \mathbb{R}^4$. In these bounds, only the third element is time-varying and defined as:

$$
\begin{aligned}
f_{\hat{y}_{3,\min}}(k) &= \begin{cases} 15, & \text{if } d_1(k_0) < 10 \\ 20, & \text{otherwise.} \end{cases}, \\
f_{\hat{y}_{3,\max}}(k) &= \begin{cases} 20, & \text{if } d_1(k_0) \geq 10 \\ 25, & \text{otherwise.} \end{cases}
\end{aligned}
\tag{18}
$$

The time varying constraints are on the indoor temperature are set such that it is colder during the night than during the day. A consequence is that during the night, when the outside temperature is colder, the controller is not working as hard to keep the temperature at a high level. Since, if there is any, the sun is heating up the greenhouse during the day, less additional heating is necessary during this period to achieve relatively high temperatures in the greenhouse hence energy can be saved [11].

III. Reinforcement Learning with economic NMPC

A powerful tool to perform optimal control without depending on a model of the system is reinforcement learning. RL methods are often based on learning the optimal control policy. Unfortunately, RL can't provide strong guarantees on the behavior of the resulting control scheme. Model predictive control (MPC), on the other hand, is a standard tool for the closed-loop control of systems with constraints and limitations. Furthermore, the theory behind (N)MPC has been developed thoroughly. In [12], it is shown that an (E)NMPC scheme can be tuned to deliver an optimal policy of the real system, even when using a wrong model. The model and (N)MPC parameters can be changed with the help of RL strategies to achieve better performance. This is seen as an opportunity in the case of greenhouse climate control and will be further explained in this chapter.

RL is a tool for tackling Markov decision processes without depending on the model underlying the state transitions. Most RL methods rely on observed state transitions and realisations of the stage cost to increase the performance of the control policy. Indirect learning methods learn an approximation of the optimal action-value function underlying the MDP. Generally, the action-value function is unknown a priori, typically a generic function, such as a deep neural network, is used to approximate it. Direct RL methods learn the optimal policy directly. Most direct methods are based on policy gradient methods. Direct RL methods often use DNN's to approximate the optimal policy and the (action-) value function associated. It can be difficult to formally analyze the closed loop behavior of systems subject to an approximate optimal policy supported by a DNN or a generic function approximation method [12], furthermore DNN's limit the amount of knowledge that can be included about the system and its underlying constraints. (E)NMPC schemes can be used instead of DNN's to support the parametrization to approximate the (action-) value functions and the policy [12]. By adapting the stage cost, terminal cost, and constraints the NMPC scheme can deliver the optimal control policy, even if the underlying model is incorrect. Using (E)NMPC as a parametrization for RL allows one to use the rich theory underlying (E)NMPC schemes in the context of RL, system knowledge and constraints can be included by using (E)NMPC as a function approximator. Most of the theory in this chapter is based on [12] [13] [14] [15]. The reader is referred to these papers for a more in depth explanation of the theory and proofs thereof.

A. Using ENMPC as a Function Approximator

A parametrized ENMPC scheme can be used to approximate the optimal policy and the value functions $\pi_\star$, $V_\star$, and $Q_\star$, even if the underlying model of the ENMPC is not accurate. System knowledge and constraints can be captured in the ENMPC scheme, while the RL algorithm will help improve the performance of the system.

The true greenhouse model can be described by a discrete MDP having the state transition dynamics:

$$\mathbb{P}[s_+ | s, a] \tag{19}$$

Which, in more typical control fashion, can also be written as:

$$s_+ = f(s, a, \zeta) \tag{20}$$

For this thesis it is assumed that the true model of the system is not known and an approximate model is taken into account for the control purposes, having the following transition dynamics:

$$\mathbb{P}[\hat{s}_+ | s, a] \tag{21}$$

This model does not necessarily match the real model (Equation 19), indicated by $\hat{s}_+$. In the approximate instance, f is replaced by $\hat{f}_\theta$.

The ENMPC scheme will be based upon the imperfect model of the greenhouse system, $\hat{f}_\theta$, and seek the minimization of the associated stage cost $\hat{L}(s,a)$.

$$\hat{L}(s,a) \;=\; \begin{cases} Q_\star(s,a) - \gamma \mathcal{V}_\star^+(s,a), & \text{if } |\mathcal{V}_\star^+(s,a)| < \infty \\ \infty, & \text{otherwise.} \end{cases} \tag{22}$$

Where $\mathcal{V}^+(s,a) = \mathbb{E}[V_\star(\hat{s}_+|s,a]$, and the expectation is taken over the approximate system model. It should be noted that the conditional formulation of the stage cost is to create a well-defined $\hat{L}$. In [12], it is stated that, under some conditions, the optimal policy $\pi_\star$ that minimizes the stage cost for the true dynamics is also generated by using the approximate model with its associated stage cost $\hat{L}$. Thus, making it possible to find the optimal policy based on a wrong model. $\hat{L}$ can be learned directly from the data.

$\hat{L}$ is not known a priori and will have to be learned, as a result the parametrization of the mixed constraints will also have to be learned through RL tools. The constraints can be used in the ENMPC scheme to exclude the undesirable states and inputs. In the most generic form the constraints will consist of pure input constraints, and mixed constraints which constrain combinations of states and inputs:

$$g(a) \;\leq\; 0, \quad h(s,a) \;\leq\; 0 \tag{23}$$

Since RL can't handle infinite values easily the domain where $\hat{L}$ is finite needs to be captured, the pure input constraints can remain fixed, but the mixed constraints need to be modified. A slack variable is added to relax the mixed constraints as follows:

$$h_\theta(x_k,u_k) \leq \sigma_k, \;\; h_\theta^f(x_N) \leq \sigma_N. \tag{24}$$

As a result, the parametrization of the value function $V_\star$ uses the following ENMPC scheme. The value function ENMPC scheme is parametrized by θ.

$$\begin{aligned} V_\theta(s) = \min_{u,x,\sigma} \; & \lambda(x_0) + \gamma^N \left(V_\theta^f(x_N) + w_f^\top \sigma_N\right) + \sum_{k=0}^{N-1} \gamma^k \left(l_\theta(x_k,u_k) + w^\top \sigma_k\right) \\ s.t. \; & x_{k+1} = f_\theta(x_k,u_k), \;\; x_0 = s \\ & g(u_k) \leq 0, \\ & h_\theta(x_k,u_k) \leq \sigma_k, \;\; h_\theta^f(x_N) \leq \sigma_N. \end{aligned} \tag{25}$$

The ENMPC scheme holds parametrizations of the model, f_θ, the mixed constraints, h_θ, the initial cost, λ_θ, the stage cost, l_θ, and the terminal cost, V_θ^f.

RL is used to increase the closed-loop performance of the system, by modifying the parameter values for the parametrization of the model, constraints and costs. The ENMPC scheme above is a classic ENMPC formulation in the case the $\gamma = 1$ and $\lambda_\theta = 0$ [16]. The mixed constraints are relaxed using slack variables, σ_k. The l_1 constraints relaxation is critical for the TD RL approaches that will be used in this work. If this relaxation were absent, the value functions might take on infinite values. Only finite values can be used for the TD learning, infinite values would cause the learning to be meaningless/infinite. The weights placed on the slack variables are denoted by w, w_f. The initial cost, λ_θ, is used to form nominal stability guarantees of the ENMPC scheme.

The action that is taken by the system is defined as taking the first element of the input sequence generated by solving the value function ENMPC scheme for a given state.

$$a = u_0^\star \tag{26}$$

Accordingly, the action-value function, Q_θ of the Q-Learning problem is given by:

$$\begin{aligned} Q_\theta(s) = \min_{u,x,\sigma} \; & \lambda(x_0) + \gamma^N \left(V_\theta^f(x_N) + w_f^\top \sigma_N\right) + \sum_{k=0}^{N-1} \gamma^k \left(l_\theta(x_k,u_k) + w \right. \\ s.t. \; & x_{k+1} = f_\theta(x_k,u_k), \;\; x_0 = s \\ & g(u_k) \leq 0, \\ & h_\theta(x_k,u_k) \leq \sigma_k, \;\; h_\theta^f(x_N) \leq \sigma_N. \\ & u_0 = a. \end{aligned} \tag{27}$$

In which only the final equality constraint is added to the scheme of Equation 25. A further difference between the two schemes, is that the cost function for the value function scheme can be slightly perturbed in order to enhance exploration. This perturbation happens by the addition of a value resulting from $f(\mathbf{u}_0, \mathbf{q}) := \rho||(\mathbf{u}_0 - \mathbf{q})||$, or $\mathbf{q}^T\mathbf{u}_0$, where $\mathbf{q}$ is randomly chosen. This parametrization will automatically satisfy the fundamental equalities underlying the Bellman equations:

$$\pi_\theta(s) = arg\min_a Q_\theta(s,a) \tag{28}$$

$$V_\theta(s) = \min_a Q_\theta(s,a) \tag{29}$$

B. RL for ENMPC

In theory, it is possible to generate the optimal policy and value functions using an ENMPC scheme based on an inaccurate model. In practice, a smart parametrization of the ENMPC scheme should be relied upon. The goal is to adjust the parameters θ in such a way that the policy resulting from the ENMPC scheme fits the optimal policy as closely as possible. The adjustment of the parameters relies on RL algortithms. In this work, classical RL approaches will be used, which are mostly gradient based. Hence, these methods require the gradient of the value functions, which requires the sensitivities of the optimal values of Q_θ [17]. The sensitivity analysis of optimization problems is detailed in [18] and delivers $\nabla_\theta Q_\theta$, needed in Equation 34. In order to evaluate the gradients of the functions Q_θ, V_θ and π_θ the Lagrange function associated with the action-value function ENMPC problem has to be found. It is observed that:

$$\nabla_\theta Q_\theta = \nabla_\theta \mathcal{L}_\theta(s, y^\star) \tag{30}$$

Where $y^\star$ holds the primal-dual variables associated to the ENMPC problem Equation 27. The Lagrangian of the action-

value function is as:

$$\begin{aligned}\mathcal{L}(s,y) =&\lambda_\theta(x_0) + \gamma^N V_\theta^f(x_N) + \chi_0^\top(x_o - s) + \mu_N^\top h_\theta^f(x_N)\\ &+ \sum_{k=0}^{N-1} \chi_{k+1}^\top(f_\theta(x_k,u_k) - x_{k+1}) + \nu_k^\top g_\theta(u_k)\\ &+ \gamma^k l_\theta(x_k,u_k) + \mu_k^\top h_\theta(x_k,u_k) + \zeta^\top(u_0 - a)\end{aligned} \tag{31}$$

Where χ, μ, ν, and ζ are the multipliers associated to the constraints of Equation 27. The gradient of Q is then created by differentiating the Lagrangian with respect to the parameters of interest.

C. Q-learning for ENMPC

One approach to Q-learning is based on parameter updates driven by the temporal difference [19]. The TD error is calculated as:

$$\tau_k = L_\theta(s_k,a_k) + \gamma V_\theta(s_{k+1}) - Q_\theta(s_k,a_k) \tag{32}$$

with gamma as the discount rate. L_θ represents the rewards function:

$$L_\theta(s_k,a_k) = l_\theta(x_k,a_k) + w^\top max(0,h_\theta(x_k,a_k)) \tag{33}$$

Where l_θ is based on the inputs and the states, and $w^\top max(0,h_\theta(x_k,a_k))$ is the importance placed on the constraint violations.

The parameter update is then given by

$$\theta \leftarrow \theta + \alpha\tau_k \nabla_\theta Q_\theta(s_k,a_k) \tag{34}$$

in which $\nabla_\theta Q_\theta(s_k,a_k)$ is found by differentiating the Lagrangian associated with the ENMPC problem of the action-value function with respect to the parameters, as explained above. The action, a_k, is selected according to the ENMPC policy. The scalar α is the step size, determining the speed at which θ is altered.

Batch updates can be used in the case of episodic tasks, such as the greenhouse climate control problem. Now, learning is performed with an alternative ENMMPC scheme. The policy is based on the original ENMPC scheme, of which the parameters can be replaced with the learned parameters at convenience. Instead of updating the parameters every time-step, the update happens at convenience. The information contained in all time-steps between parameters can be combined and used for the next parameter update. In the case of episodic tasks, the system behavior might change during an episode and system updates based on partial information can deteriorate performance.

In the remainder of this thesis Q-learning for ENMPC will be named Q-ENMPC.

IV. Set-up

A. The ENMPC scheme

The basis of the ENMPC scheme that is used in this work is the same as the one used in [10]. Some adjustments are made to the cost function and constraints, with respect to their work. In this section the changes in the ENMPC scheme of this work with respect to [10] will be pointed out.

1) Cost function: The cost function for the action-value function is constructed by including a cost on all the inputs of the system and rewarding the yield. The cost function is implemented as follows:

$$\begin{aligned}J(s) =&\lambda_\theta\\ &+ \sum_{k=0}^{N-1}\sum_{i=1}^{3} c_{u,i} * u_{i,k}\\ &+ \sum_{k=1}^{N-1} c_{y,1} * (y_k - y_{k-1})\\ &+ c_{y,2} * y_N\\ &+ \sum_{k=1}^{N} w * \sigma_k\end{aligned} \tag{35}$$

The initial cost, λ_θ, is initialised at 100. The inputs are penalised per time-step in the prediction-horizon. The three different inputs are weighted differently by $c_{u,i}$. The yield is weighted via $c_{y,1}$ and $c_{y,2}$. Lastly, the slack variables, which register constraint violations, are weighted by w. The value of the weights $c_{u,i}$, $c_{y,i}$ and w will be learned. Furthermore, to reward an increase in yield, the yield related weighting is initialized negative. Via the third term in Equation 35, part of the cost on the yield is placed on the terminal yield within the time horizon, weight is also placed on the increase (or decrease) per time-step in the prediction horizon, via the second term in Equation 35.

Exploration of the system can be enhanced by using a perturbation in the cost function used for approximating the value function:

$$V_\theta(s) = Q_\theta(s) + \mathbf{q}^\mathrm{T}\mathbf{u}_0 \tag{36}$$

Where $\mathbf{q}$ is a weight placed on the first input in the prediction horizon. This weight has a uniform distribution with mean 0. With $\mathbf{q}$ the exploration is regulated. It has been concluded that exploration is not beneficial for this research, as such $\mathbf{q}$ is set to zero.

2) Constraints: The constraints follow the structure of the model discussed in section II. It should be noted that the inequality constraints are extended with slack variables and an extra constraint on the action-value function that is shown in Equation 27. The first input of the action-value function, Q_θ, is fixed to the first input resulting from calculating the value-function, V_θ.

The objective of the final optimization problem is given by Equation 35. The constraints are given in Equation 12. Slack variables are added to the inequality constraints, similarly to Equation 27. A final constraint is added with the addition of the final equality constraint of Equation 27.

B. Changing parameters

For the various case studies, a couple of factors are of importance to make the system learn correctly. In the following, these factors will be discussed briefly.

1) RL reward function: The reward function of the RL agent is the main differentiator in the learning process. It dictates what is good progress and what is bad behavior of the system. The main parts that will be present in the reward function are a rewards for the performance, and a penalty for constraint violations.

c_1 and c_2 are the weights placed on the performance and the constraint violations, allowing one to decide what is important in the learning process. Generally, the value of c_1 will be between zero and a negative integer, while the value of c_2 will be between 0 and a positive integer. The reward function is part of the TD error as described in subsection III-C. Hence the reward function is defined as:

$$L_\theta(s_k, a_k) = c_1 * l_\theta(x, a) + c_2 * \sum_{k=1}^{N} (\max\{0, h_\theta(x_k, a_k)\}) \tag{37}$$

Where $l_\theta(x, a)$ is taken as the EPI, explained in subsection IV-C, minus a fixed integer, 1.8. $c_2 * \sum_{k=1}^{N} (\max\{0, h_\theta(x_k, a_k)\})$ is the sum of the magnitude of the constraint violations at each time-step, as discussed in subsection IV-C.

2) Exploration: Exploration is done through the addition of a perturbation to the value function in the ENMPC scheme. This perturbation is explained in further detail in subsection III-A. In this work, no exploration will be included.

3) Learning rate: The learning rate determines the rate at which the parameter values are adjusted during learning. High learning rates can cause the system to 'jump' around the optimal values or even cause instability, while too low of a learning rate causes slower learning than necessary.

The learning rates are initiated to cause a 1% change in the values of the model and ENMPC parameters. The learning rate for the value of weight on the slack variables is higher, at a rate of 50%. The learning rates are decreased by 10% every episode during learning.

4) Slack variables: Slack variables are needed for Q-ENMPC to work, as described in section III. The weights placed on the slack variables are an important factor in determining how conservative the system is and in decreasing the constraint violations. The initialization of the slack variables is a factor in the initial constraint violations.

Since the learning rate is a percentage of the initial value of the parameters, the weight on the slack variables should not be too low, as this could prohibit relevant learning of the parameter. A value of 5000 is settled upon as good starting ground for learning, while not yet prohibiting constraint violations so much that none are present at the start.

5) Parameter initialization: Multiple different initializations for the values of the model parameters are used in the case studies. First, the nominal values for the model parameters can be used. When the values for the model parameters are initialized with their nominal value, it will be referred to as nominal initialization.

Second, in some case studies the parameters are initialised within a band around the nominal parameters of 20%, as explained in subsection II-A. This means that the value for each parameter deviates a maximum of 20% from the nominal value. The actual deviation is chosen at random, such that a significant increase in constraint violations is present, compared to the nominal parameters. When the values for the model parameters are initialized with this random value, it will be referred to as randomized initialization.

The model parameters values with which the system is initialised in the case of randomized initialization are presented in Table I.

parameter	value	parameter	value
$c_{1,1}$	0.37354	$c_{2,1}$	3.459
$c_{1,2}$	$2.242 \cdot 10^{-7}$	$c_{2,2}$	$4.434 \cdot 10^{-7}$
$c_{1,3}$	50.69	$c_{2,3}$	$7.5 \cdot 10^{-6}$
$c_{1,4}$	$2.07 \cdot 10^{-9}$	$c_{2,4}$	75560
$c_{1,5}$	$4.52 \cdot 10^{-6}$	$c_{2,5}$	273.15
$c_{1,6}$	$3.65 \cdot 10^{-4}$	$c_{2,6}$	101325
$c_{1,7}$	$5.96 \cdot 10^{-4}$	$c_{2,7}$	0.044
$c_{1,8}$	$5.2 \cdot 10^{-5}$		
parameter	value	parameter	value
$c_{3,1}$	$2.52 \cdot 10^{4}$	$c_{4,1}$	4.799
$c_{3,2}$	1070	$c_{4,2}$	0.0032
$c_{3,3}$	6.03	$c_{4,3}$	9348
$c_{3,4}$	0.207	$c_{4,4}$	8314
		$c_{4,5}$	273.15
		$c_{4,6}$	17.4
		$c_{4,7}$	239
		$c_{4,8}$	17.269
		$c_{4,9}$	238.3

TABLE I: Initialisation values of the model parameters.

Thirdly, an initialization is used in which the value of all model parameters is initialized at 1.1 times their nominal value. When the values for the model parameters are initialized in this way, it will be referred to as initialization 1.1.

Lastly, a more complicated initialization is used. Initialization 1.1 is used as a starting point. This initialization is then used to learn for constraint violations, the reward function and results are shown in case study 2. By learning for constraint violations, the EPI decreases rapidly within a limited number of episodes. The parameters of the system in the third episode are used for the this initialization. This initialization will only be used in case study 1, it will be referred to as initialization 3. The benefit of this initialization is that it is known beforehand that in a limited amount of parameter updates a significantly increased EPI can be achieved.

Five parameters are usually kept to their true value, this is done because they represent well known physical constants (e.g. conversion factor from Kelvin to Celsius). In one cases, while using a randomized initialization these values are also randomized, in that case the initialization will be referred to as randomized$_\text{r}$.

The initial value of the learn-able parameters of the ENMPC scheme are presented in Table II. In the table, it can be seen that the inputs are penalised, while the yield is rewarded. The initial value of the learn-able parameters of the ENMPC scheme is the same for all case studies.

Parameter	Initial value
Initial cost	100
cost on CO_2 input	50
cost on ventilation	1
cost on heating input	1000
cost on yield increase	-10
cost on terminal yield	-100
Weight on slack variables	5000

TABLE II: Initialisation parameters of the ENMPC scheme.

6) RL updates: During training parameter values for the CO_2 capacity of the greenhouse, heat capacity of the greenhouse air and coefficient of leaf-air vapor flow, for the CO_2 concentration, temperature and humidity are updated by the RL agent. The interval between updates is a hyperparatemer in the controller design. In this study the parameters will be updated once per episode, with an episode being the length of one growth cycle. It is chosen to only update once per episode because the gradient of the system changes significantly during the episode. It would not be feasible to implement a changing learning rate. Due to there only being one update per episode, it is the amount of episodes that determines the amount of parameter updates.

The information contained in all time-steps is combined to form the TD error for the parameter update. As discussed in section III, the TD error for each episode is given by

$$\tau_k = L_\theta(s_k, a_k) + \gamma V_\theta(s_{k+1}) - Q_\theta(s_k, a_k) \tag{38}$$

with the rewards function defined in Equation 37. As explained in subsection III-C, the parameter update is then given by

$$\theta \leftarrow \theta + \alpha \tau_k \nabla_\theta Q_\theta(s_k, a_k) \tag{39}$$

The reward function is calculated once per episode, and changes for each case study. The gradient is averaged over the entire episode for each parameter. The average gradient is then used in the update equation above.

C. Performance indicators

The first performance indicator used in this work was introduced in [20], an economic profit indicator. The economic profit indicator calculates the profit that stems from a certain growth cycle of lettuce, when the lettuce is harvested and sold at the end of that cycle. The function is as follows:

$$EPI = \phi(y_1(t_\mathrm{f})) - \sum_{t_\mathrm{b}}^{t_\mathrm{f}} (c_\mathrm{q} u_\mathrm{q}(t) + c_{\mathrm{CO}_2} u_{\mathrm{CO}_2}(t)) dt \tag{40}$$

where $\phi(y_1(t_\mathrm{f}))$ is the gross income from selling the harvested lettuce, which is obtained at harvest time t_f, in $Hflm^{-2}$. The operating cost of the airconditioning equipment is indicated by $c_\mathrm{q} u_\mathrm{q}(t) + c_{\mathrm{CO}_2} u_{\mathrm{CO}_2}(t)$. The auction price of the lettuce is assumed to follow a linear ratio $\phi(y_1(t_\mathrm{f})) = c_{\mathrm{pri},1} + c_{\mathrm{pri},2} y_1(t_\mathrm{f})$ between the auction price and harvest weight. $c_{\mathrm{pri},1}$ in $Hflm^{-2}$ and $c_{\mathrm{pri},2}$ in $Hflkg^{-1}m^{-2}$. It is assumed that the operating cost of the air conditioning equipment are related to the amount of energy, u_q in (Wm^{-2}), and the amount of carbon dioxide introduced into the system, u_{CO_2} in $(kgm^{-2}s^{-1}$, linearly. The operating costs are parameterised by the price of energy, c_q in $(HflJ^{-1})$, and the price of carbon dioxide, c_{CO_2} in $(Hflkg^{-1})$. The values of these parameters can be found in Table III.

parameter	value
c_{co_2}	$42 \times 10^{-2}\ Hflkg^{-1}$
c_q	$6.35 \times 10^{-9}\ HflJ^{-1}$
$c_{\mathrm{pri},1}$	$1.8\ Hflm^{-2}$
$c_{\mathrm{pri},2}$	$16\ Hflm^{-1}$

TABLE III: Parameters of the economic profit function.

The second performance indicator is the constraint violations during the growth cycle. The constraint violations are calculated as the total amount of constraint violation summed over the entire horizon. For example, the humidity constraint is always set at 70%. If, during three time-steps the humidity is at 75%, the total constraint violation is 15. During those same time-steps it could be that the temperature constraint is also violated by one degree, then the total constraint violation is 15, for the humidity violation, plus 3, for the temperature violation, for a total of 18. The total constraint violations over the growth cycle are divided by 50 to reach a more intuitive number.

V. Numerical simulation

The aim of this case study is to decrease constraint violations over time. In this case, performance based on EPI is not the main concern. The first interesting case is to initialize the system with the true parameters and try to find parameter updates that will decrease the constraint violations from there, ideally with only a minor decrease in EPI. The second case is to decrease constraint violations from the parameters that result in maximum EPI, these parameters cause quite significant constraint violations that could be harmful for the health of the crop.

In this case study the parameters are initializes in multiple ways. First, the randomized initialization is used. Secondly, initialization 1.1 is used. Lastly, simulations starting from the nominal model parameter values have been done, unfortunately some optimization error occurred that could not be solved within the time available.

The RL reward function is kept the same in both simulations:

$$\text{RL reward} = 0.04 * \sum_{k=1}^{N} (\max\{0, h_\theta(x_k, a_k)\}) \tag{41}$$

The factor for c_2, 0.04, is kept small such that the contribution of the constraint violations to the TD error is of the same order of magnitude as the difference between the value functions.

A. Results

In Figure 2 and Figure 3, the EPI and constraint violations of the system are shown while learning for minimizing constraint violations starting from the randomized initialization. The results indicate that a significant decrease in constraint violations is possible when starting from these initial parameter values. The best results are achieved when using initialization 1.1. The initial constraint violations are at a level over 504, while

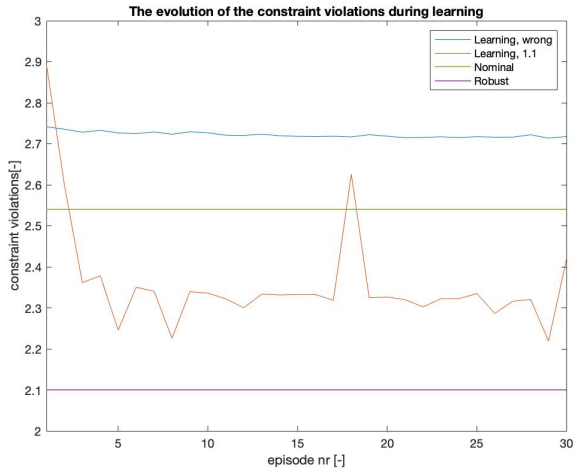

Fig. 2: The evolution of the economic performance during learning for the constraint violations. In blue the evolution of the EPI is shown during learning, when the parameters are initialized as in Table I. In orange the economic performance is shown when the parameters are initialized as 1.1 times the nominal value. In green the performance with nominal parameter values, and in purple the performance with samble-based MPC with 10 samples and 10% uncertainty. It can be seen that the economic performance stays close to its initial level in the first case, always being higher than in both the nominal and the robust case. In the second case, the performance drops significantly, but stays higher than in the robust case.

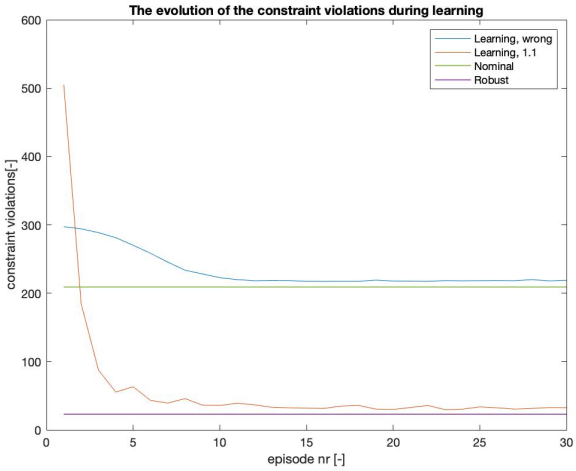

Fig. 3: The evolution of the constraint violations during learning for the constraint violations. In blue the evolution of the constraint violations is shown during learning, when the parameters are initialized as in Table I. In orange the evolution of constraint violations is shown when the parameters are initialized as 1.1 times the nominal value. In green the constraint violations with nominal parameter values, and in purple the constraint violations with samble-based MPC with 10 samples and 10% uncertainty. It can be seen that the constraint violations drop in both cases, but much more significantly in the second case.

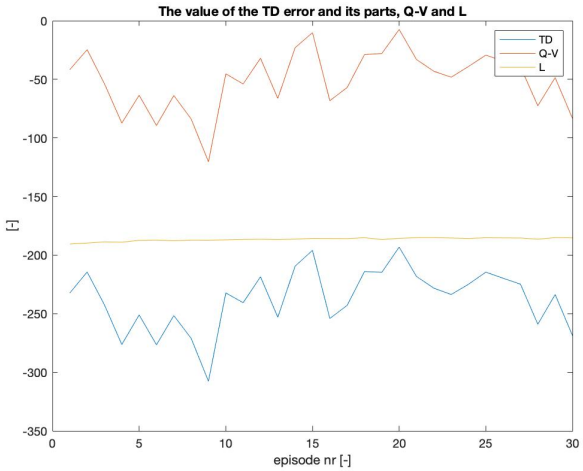

Fig. 4: The evolution of the TD error during learning for the constraint violations. The parameters are initialized as in Table I. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions. It can be seen that the reward function converges to around -185, while the difference between the value functions stays around the same level.

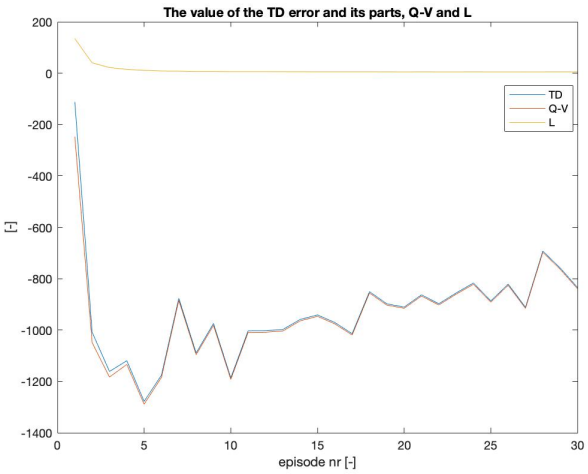

Fig. 5: The evolution of the TD error during learning for the constraint violations. The parameters are initialized as 1.1 times the nominal parameter value. In blue the evolution of the TD error is shown during learning, in yellow the evolution of the reward function is shown, and in orange the difference between the value functions. It can be seen that the reward function converges to around 5, while the difference between the value functions first drops and then creeps up.

the final level is around 30, a decrease of 94% and well under the level achieved with the nominal parameter values.

Using the randomized initialization also results in a significant improvement of the constraint violations, but the improvement is not as good as with initialization 1.1. The initial constraint violation is at 297, and the final level is around 218, slightly above the level of constraint violations with the nominal parameter values. This decrease is about 26.6%.

The simulations show that the EPI decreases during learning. The EPI decreases from 2.73 to around 2.71 when the randomized initialization is used. When initialization 1.1 is used, the decrease is from 2.89, to around 2.3, a decrease of 20.5%.

The results achieved initialization 1.1 are notable because they are able to achieve constraint violations close to that of robust MPC, while still having economic performance almost 10% higher. The slack variables were expected to play the largest role in decreasing the constraint violations, by increasing the weight on the slack variables it is more costly to violate constraints. It is interesting to see that the weight on the slack variables is increased more in the case that the randomized initialization is used. In the randomized initialization case the weight increases to $3e4$, while in the initialization 1.1 case the weight is only increased to $2e4$.

Both cases do not show convergence of the TD error. This lack of convergence can mainly be attributed to the behavior of the value functions. The value of the TD error is dominated by the value functions.

Starting from the nominal parameter values has presented some unexpected issues. Due to unknown solver issues, the results stagnate after around 2000 time-steps and become unusable. Multiple scenarios were investigated, but unfortunately the issue could not be resolved in time.

Constraint violations are only seen in the temperature and humidity constraints. The temperature bounds are slightly violated both on the high end and low ends, this is deemed to little to be of concern. The humidity is constantly being violated on the high end. It is quite logical to steer towards violating this bound, as a high humidity helps with crop growth according to the model. The bounds are in place to prevent crop disease, which is not explicitly accounted for in the model. Literature reports higher chances of crop disease at elevated humidity levels, however, studies are conducted at humidity levels of 80% or higher, which is significantly higher than the average learned humidity level of 71%. It can also be seen that at the start of the learning process the average humidity is at over 73%, which is significantly higher than the average humidity at the end of learning.

VI. Conclusions

In this paper, a data-driven ENMPC using reinforcement learning approach is proposed to tackle the greenhouse climate control problem. A framework is constructed in which ENMPC functions as the function approximator within a Q-learning framework. Within this framework the ENMPC applies the best action for the known system to the real system, while the Q-learning agent alters the parameters of the ENMPC control structure to better fit the real system. The RL reward function is constructed to punish constraint violations. The mpc parameters are updated once per growth cycle of the crop.

A simulation-based case study is performed in which the economic performance and constraint violations of the Q-ENMPC are compared to that of nominal ENMPC to demonstrate the effectiveness of Q-ENMPC. Starting from the perturbed initial model parameter values, the system reaches an economic performance 10% higher than the system controlled with robust MPC, while having constraint violations that are similar. As it is currently unclear what performance is achieved in commercial greenhouses, the question remains what the theoretical change in performance is with respect to commercial circumstances. Learning would take a significant amount of time were it to be done in the real world. Since one episode encompasses a full growth cycle of the lettuce crop, and the systems approaches its final performance only after 10 episodes, learning would take over a year in the real world.

References

[1] G. V. Straten and E. J. V. Henten, "Optimal greenhouse cultivation control: Survey and perspectives," vol. 3, 2010.

[2] E. van Henten, "Monitoring and control or greenhouse crop production," 2013.

[3] M. Taki, A. Rohani, and M. Rahmati-Joneidabad, "Solar thermal simulation and applications in greenhouse," *Information Processing in Agriculture*, vol. 5, pp. 83–113, 3 2018.

[4] N. van der Velden and P. Smit, *Energiemonitor van de Nederlandse glastuinbouw 2018*. No. 2019-111 in Wageningen Economic Research rapport, Wageningen Economic Research, 2019. Project code 2282200493.

[5] H. Challa and G. van Straten, "Reflections about optimal climate control in greenhouse cultivation," *IFAC Proceedings Volumes*, vol. 24, pp. 13–18, 9 1991.

[6] T. H. Oh, H. M. Park, J. W. Kim, and J. M. Lee, "Integration of reinforcement learning and model predictive control to optimize semi-batch bioreactor," *AIChE Journal*, vol. 68, p. e17658, 6 2022.

[7] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, "Fusion of machine learning and mpc under uncertainty: What advances are on the horizon?," 2022.

[8] Q. Zhang, W. Pan, and V. Reppa, "Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 7 2021.

[9] E. van Henten, "Greenhouse climate management: An optimal control approach," 1994.

[10] S. Boersma, C. Sun, and S. V. Mourik, "Robust sample-based model predictive control of a greenhouse system with parametric uncertainty," 2022.

[11] S. Boersma and S. V. Mourik, "Sample-based nonlinear model predictive control in a lettuce greenhouse with uncertain weather forecasting," 2021.

[12] S. Gros and M. Zanon, "Reinforcement learning for mixed-integer problems based on mpc," *IFAC-PapersOnLine*, vol. 53, pp. 5219–5224, 1 2020.

[13] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, vol. 65, 2020.

[14] S. Gros and M. Zanon, *Reinforcement Learning based on MPC and the Stochastic Policy Gradient Method; Reinforcement Learning based on MPC and the Stochastic Policy Gradient Method*. 2021.

[15] M. Zanon and S. Gros, "Safe reinforcement learning using robust mpc; safe reinforcement learning using robust mpc," *IEEE Transactions on Automatic Control*, vol. 66, 2021.

[16] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2020.

[17] H. N. Esfahani, A. B. Kordabad, and S. Gros, "Approximate robust nmpc using reinforcement learning," pp. 132–137, Institute of Electrical and Electronics Engineers Inc., 2021.

[18] C. Büskens and H. Maurer, "Sensitivity analysis and real-time optimization of parametric nonlinear programming problems," 2001.
[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*. Bradford Books, second edition ed., 2018.
[20] B. Morcego, W. Yin, S. Boersma, E. van Henten, V. Puig, and C. Sun, "Reinforcement learning versus model predictive control on greenhouse climate control," *Computers and Electronic in Agriculture*, 2023.

Bibliography

[1] U. Deichmann, A. Goyal, and D. Mishra, "Will digital technologies transform agriculture in developing countries?," *Agricultural Economics*, 2016.

[2] WUR, "Internationaal - wur."

[3] G. V. Straten and E. J. V. Henten, "Optimal greenhouse cultivation control: Survey and perspectives," vol. 3, 2010.

[4] E. van Henten, "Monitoring and control or greenhouse crop production," 2013.

[5] M. Taki, A. Rohani, and M. Rahmati-Joneidabad, "Solar thermal simulation and applications in greenhouse," *Information Processing in Agriculture*, vol. 5, pp. 83–113, 3 2018.

[6] N. van der Velden and P. Smit, *Energiemonitor van de Nederlandse glastuinbouw 2018*. No. 2019-111 in Wageningen Economic Research rapport, Wageningen Economic Research, 2019. Project code 2282200493.

[7] H. Challa and G. van Straten, "Reflections about optimal climate control in greenhouse cultivation," *IFAC Proceedings Volumes*, vol. 24, pp. 13–18, 9 1991.

[8] T. H. Oh, H. M. Park, J. W. Kim, and J. M. Lee, "Integration of reinforcement learning and model predictive control to optimize semi-batch bioreactor," *AIChE Journal*, vol. 68, p. e17658, 6 2022.

[9] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, "Fusion of machine learning and mpc under uncertainty: What advances are on the horizon?," 2022.

[10] Q. Zhang, W. Pan, and V. Reppa, "Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 7 2021.

[11] D. Katzin, E. J. van Henten, and S. van Mourik, "Process-based greenhouse climate models: Genealogy, current status, and future directions," *Agricultural Systems*, vol. 198, 2022.

[12] P. Ponce, A. Molina, P. Cepeda, E. Lugo, and B. MacCleery, *Greenhouse Design and Control*. 2015.

[13] K. Nemali, "History of controlled environment horticulture: Greenhouses," *HortScience*, vol. 57, pp. 239–246, 2 2022.

[14] L. Marcelis, J. Costa, and E. Heuvelink, "Achieving sustainable greenhouse production: present status, recent advances and future developments," *Achieving Sustainable Greenhouse Cultivation*, pp. 1–12, 2019.

[15] O. Körner, "Models, sensors and decision support systems in greenhouse cultivation," *Advances in crop modelling for a sustainable agriculture*, pp. 379–403, 2019.

[16] H. G. Management, "Oogstprognose - hoogendoorn growth management," 2022.

[17] Priva, "Tuinbouw automatisering | water & klimaatbeheersing | priva," 2022.

[18] B-Mex, "B-mex," 2021.

[19] S. Hemming, F. D. Zwart, A. Elings, I. Righini, and A. Petropoulou, "Remote control of greenhouse vegetable production with artificial intelligence—greenhouse climate, irrigation, and crop production," *Sensors 2019, Vol. 19, Page 1807*, vol. 19, p. 1807, 4 2019.

[20] X. Cao, Y. Yao, L. Li, W. Zhang, Z. An, Z. Zhang, S. Guo, L. Xiao, X. Cao, and D. Luo, "Igrow: A smart agriculture solution to autonomous greenhouse control," 4 2021.

[21] B. A. Keating and P. J. Thorburn, "Modelling crops and cropping systems—evolving purpose, practice and prospects," *European Journal of Agronomy*, vol. 100, pp. 163–176, 10 2018.

[22] J. Thornley and J. France, "Mathematical models in agriculture: Quantitative methods for the plant ... - j. h. m. thornley, j. france - google boeken," 2007.

[23] W. Duncan, L. Maize, and T. Evans, "Crop physiology: Some case histories - google boeken," 1975.

[24] J. C. Roy, T. Boulard, C. Kittas, and S. Wang, "Pa—precision agriculture: Convective and ventilation transfers in greenhouses, part 1: the greenhouse considered as a perfectly stirred tank," *Biosystems Engineering*, vol. 83, pp. 1–20, 9 2002.

[25] H. F. DeZwart, "Determination of direct transmission of a multispan greenhouse using vector algebra," *Journal of Agricultural Engineering Research*, vol. 56, pp. 39–49, 9 1993.

[26] N. Choab, A. Allouhi, A. E. Maakoul, T. Kousksou, S. Saadeddine, and A. Jamil, "Review on greenhouse microclimate and application: Design parameters, thermal modeling and simulation, climate controlling technologies," *Solar Energy*, vol. 191, pp. 109–137, 10 2019.

[27] M. S. Ahamed, H. Guo, and K. Tanino, "Modeling heating demands in a chinese-style solar greenhouse using the transient building energy simulation model trnsys," *Journal of Building Engineering*, vol. 29, p. 101114, 5 2020.

[28] T. Boulard, C. Kittas, J. C. Roy, and S. Wang, "Se—structures and environment: Convective and ventilation transfers in greenhouses, part 2: Determination of the distributed greenhouse climate," *Biosystems Engineering*, vol. 83, pp. 129–147, 10 2002.

[29] E. van Henten, "Greenhouse climate management: An optimal control approach," 1994.

[30] H. F. D. Zwart, "Analyzing energy-saving options in greenhouse cultivation using a simulation model," 1996.

[31] E. J. V. Henten, "Sensitivity analysis of an optimal control problem in greenhouse climate management," *Biosystems Engineering*, vol. 85, 2003.

[32] E. J. V. Henten and J. Bontsema, "Time-scale decomposition of an optimal control problem in greenhouse climate management," *Control Engineering Practice*, vol. 17, 2009.

[33] C. Spitters, H. van Keulen, and D. van Kraalingen, "A simple and universal crop growth simulator: Sucros87," 1989.

[34] D. J. Sweeney, D. W. Hand, G. Slack, and J. H. M. Thornley, "Modelling the growth of winter lettuce," *Mathematics and Plant Physiology in Experimental Botany*, vol. 16, pp. 217–229, 1981.

[35] B. A. Bouman, H. V. Keulen, H. H. V. Laar, and R. Rabbinge, "The 'school of de wit' crop growth simulation models: A pedigree and historical overview," *Agricultural Systems*, vol. 52, pp. 171–198, 10 1996.

[36] F. I. Soribe and R. B. Curry, "Simulation of lettuce growth in an air-supported plastic greenhouse," *Journal of Agricultural Engineering Research*, vol. 18, pp. 133–140, 6 1973.

[37] T. R. Wheeler, P. Hadley, J. I. Morison, and R. H. Ellis, "Effects of temperature on the growth of lettuce (lactuca sativa l.) and the implications for assessing the impacts of potential climate change," *European Journal of Agronomy*, vol. 2, pp. 305–311, 1 1993.

[38] S. Pearson, T. R. Wheeler, P. Hadley, and A. E. Wheldon, "A validated model to predict the effects of environment on the growth of lettuce (lactuca sativa l.): Implications for climate change," *http://dx.doi.org.tudelft.idm.oclc.org/10.1080/14620316.1997.11515538*, vol. 72, pp. 503–517, 1997.

[39] D. L. Critten, "Optimization of co2 concentration in greenhouses: A modelling analysis for the lettuce crop," *Journal of Agricultural Engineering Research*, vol. 48, pp. 261–271, 1 1991.

[40] G. Ascher, "Optimization of carbon dioxide and temperature control for greenhouse crop production using models of different complexity," 1993.

[41] A. J. Both, "Dynamic simulation of supplemental lighting for greenhouse hydroponic lettuce production," 1995.

[42] I. Seginer, G. Shina, L. D. Albright, and L. S. Marsh, "Optimal temperature setpoints for greenhouse lettuce," *Journal of Agricultural Engineering Research*, vol. 49, pp. 209–226, 5 1991.

[43] I. Seginer, G. V. Straten, and F. Buwalda, "Nitrate concentration in greenhouse lettuce: A modeling study," *Acta Horticulturae*, vol. 456, pp. 189–197, 1998.

[44] I. Seginer, G. V. Straten, and F. Buwalda, "Lettuce growth limited by nitrate supply," *Acta Horticulturae*, vol. 507, pp. 141–148, 1999.

[45] I. Seginer, "A dynamic model for nitrogen-stressed lettuce," *Annals of Botany*, pp. 623–635, 2003.

[46] I. Seginer, "Equilibrium and balanced growth of a vegetative crop," 2004.

[47] I. Seginer, P. Bleyaert, and M. Breugelmans, "Modelling ontogenetic changes of nitrogen and water content in lettuce," *Annals of Botany*, vol. 94, pp. 393–404, 9 2004.

[48] S. Boersma and S. V. Mourik, "Sample-based nonlinear model predictive control in a lettuce greenhouse with uncertain weather forecasting," 2021.

[49] S. Boersma, C. Sun, and S. V. Mourik, "Robust sample-based model predictive control of a greenhouse system with parametric uncertainty," 2022.

[50] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2020.

[51] R. Babuska and J. Kober, *Knowledge-Based Control Systems*. Delft University of Technology, 3 2021.

[52] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*. Bradford Books, second edition ed., 2018.

[53] S. Gros and M. Zanon, "Reinforcement learning for mixed-integer problems based on mpc," *IFAC-PapersOnLine*, vol. 53, pp. 5219–5224, 1 2020.

[54] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, vol. 65, 2020.

[55] S. Gros and M. Zanon, *Reinforcement Learning based on MPC and the Stochastic Policy Gradient Method; Reinforcement Learning based on MPC and the Stochastic Policy Gradient Method*. 2021.

[56] M. Zanon and S. Gros, "Safe reinforcement learning using robust mpc; safe reinforcement learning using robust mpc," *IEEE Transactions on Automatic Control*, vol. 66, 2021.

[57] H. N. Esfahani, A. B. Kordabad, and S. Gros, "Approximate robust nmpc using reinforcement learning," pp. 132–137, Institute of Electrical and Electronics Engineers Inc., 2021.

[58] C. Büskens and H. Maurer, "Sensitivity analysis and real-time optimization of parametric nonlinear programming problems," 2001.

[59] E. J. V. Henten, "Validation of a dynamic lettuce growth model for greenhouse climate control," *Agricultural Systems*, vol. 45, 1994.

[60] B. Morcego, W. Yin, S. Boersma, E. van Henten, V. Puig, and C. Sun, "Reinforcement learning versus model predictive control on greenhouse climate control," *Computers and Electronic in Agriculture*, 2023.

Glossary

List of Acronyms

CFD	Computational Fluid Dynamics
FIR	Far Infrared
MDP	Markov Decision Process
NiCoLet	Nitrate Control in Lettuce
SUCROS87	Simple and Universal Crop Growth Simulator

List of Symbols

Symbols

α	Learning rate
$\hat{p}$	uncertain parameters
λ	Initial cost
$\mathcal{L}_\theta$	Lagrangian, influenced by theta
σ	slack variable
τ_{k}	TD error at time k
$\mathbf{q}$	Value determining exploration magnitude
θ	Parameters subject to learning
c_1	Weight placed on the EPI in the RL reward function
c_2	Weight placed on the constraint violations in the RL reward function
C_{Inj}	CO_2 injected by control equipment
C_{Phot}	CO_2 flow due to photosynthesis
C_{Vent}	CO_2 flow due to ventilation
d_1	disturbance state, radiation
d_2	disturbance state, outdoor CO_2 concentration

d_3	disturbance state, outdoor temperature
d_4	disturbance state, outdoor humdity
E	Energy in the greenhouse
h_θ	Inequality constraints, influenced by parameters theta
k	discrete time of the system
$L_\theta(s)$	RL reward function
l_θ	stage cost, influenced by parameters theta
M_{C}	CO_2 concentration in the greenhouse
M_{W}	Water vapor mass in the greenhouse
N_{p}	Prediction horizon taken into account MPC
N_{s}	Number of samples taken into account for robust control
p	parameters
Q_{Con}	Convective and conductive heat exchange with outside climate
Q_{Cool}	Heat input from coolers
Q_{FIR}	Heat input from far infrared radiation
Q_{Heat}	Heat input from heaters
Q_{Lamp}	Heat input from lamps
Q_{Latent}	conversion from sensible to latent heat
Q_{Sun}	Heat input from the sun
Q_{Vent}	Heat input through ventilation
$Q_\theta(s)$	Action-value function
S_{i}	Sensitivity of the dry weight to parameter i
t_{b}	Time at beginning of growth cycle
t_{f}	Time of harvest
u_0	Action resulting from the calculation of the value function
u_1	Input, CO_2 injection
u_2	Input, ventilation
u_3	Input, heat input
$V_\theta(s)$	Value function
V_θ^{f}	Terminal cost
w	Weight on the slack variables
W_{Cond}	Water vapor mass flow due to condensation
W_{Dehum}	Water vapor mass flow due to dehumidification
W_{Evap}	Water vapor mass flow due evaporation
W_{Hum}	Water vapor mass flow due to humidification
W_{Trans}	Water vapor mass flow due to crop transpiration
W_{Vent}	Water vapor mass flow due to ventilation
x_1	state, crop dry weight
x_2	state, indoor CO_2 concentration
x_3	state, indoor temperature

x_4	state, indoor humidity
X_{c}	State, greenhouse CO_2 concentration
X_{d}	State, crop dry weight
X_{h}	State, greenhouse humidity
X_{T}	State, greenhouse temperature
Y_{dw}	crop dry weight in the sensitivity function