LUND UNIVERSITY

Learning-Based Controller Design with Application to a Chiller Process

Rosdahl, Christian

2022

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Rosdahl, C. (2022). *Learning-Based Controller Design with Application to a Chiller Process*. [Licentiate Thesis, Department of Automatic Control]. Department of Automatic Control, Lund Institute of Technology, Lund University.

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

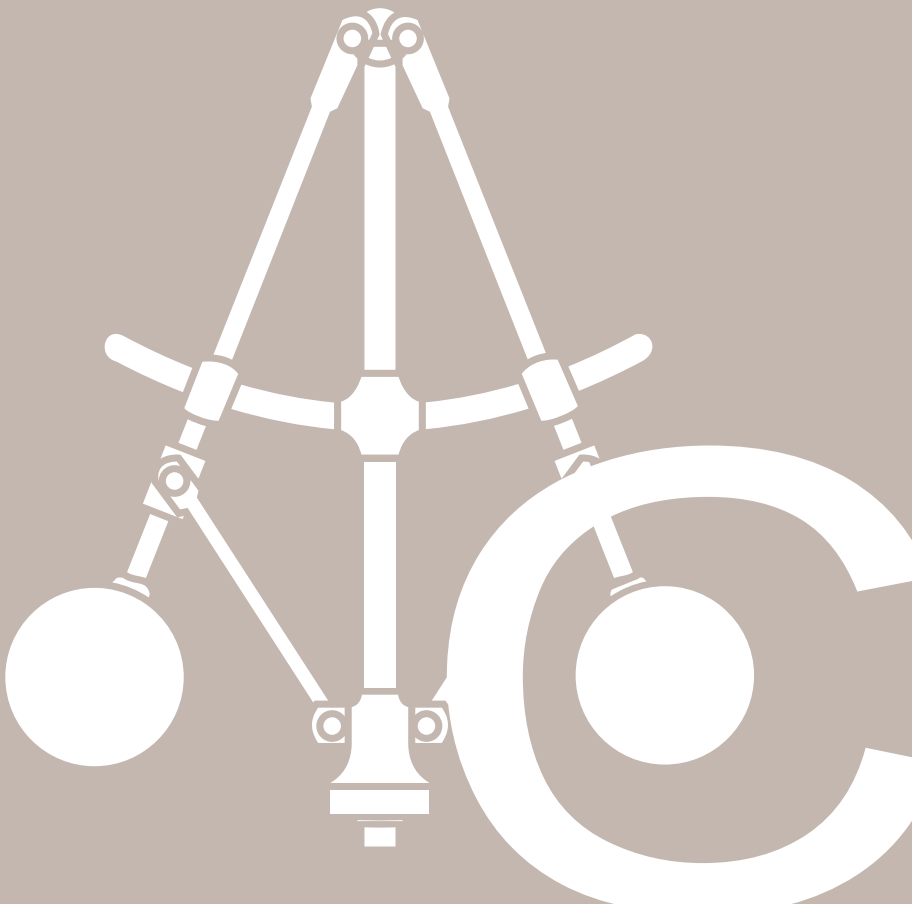
Learning-Based Controller Design with Application to a Chiller Process

CHRISTIAN ROSDAHL
DEPARTMENT OF AUTOMATIC CONTROL | LUND UNIVERSITY

Printed by Media-Tryck, Lund 2022 NORDIC SWAN ECOLABEL 3041 0903

Department of Automatic Control
P.O. Box 118, 221 00 Lund, Sweden
www.control.lth.se

ISRN LUTFD2/TFRT--3276--SE
ISSN 0280–5316



Learning-Based Controller Design with Application to a Chiller Process

Christian Rosdahl

LUND
UNIVERSITY

Department of Automatic Control

Licentiate Thesis TFRT-3276
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2022 by Christian Rosdahl. All rights reserved.
Printed in Sweden by Media-Tryck.
Lund 2022

Abstract

In this thesis, we present and study a few approaches for constructing controllers for uncertain systems, using a combination of classical control theory and modern machine learning methods. The thesis can be divided into two subtopics. The first, which is the focus of the first two papers, is dual control. The second, which is the focus of the third and last paper, is multiple-input multiple-output (MIMO) control of a chiller process.

In dual control, the goal is to construct controllers for uncertain systems that in expectation minimize some cost over a certain time horizon. To achieve this, the controller must take into account the dual goals of accumulating more information about the process, by applying some probing input, and using the available information for controlling the system. This is referred to as the exploration-exploitation trade-off. Although optimal dual controllers in theory can be computed by solving a functional equation, this is usually intractable in practice, with only some simple special cases as exceptions. Therefore, it is interesting to examine methods for approximating optimal dual control.

In the first paper, we take the approach of approximating the value function, which is the solution of the functional equation that can be used to deduce the optimal control, by using artificial neural networks. In the second paper, neural networks are used to represent and estimate hyperstates, which contain information about the conditional probability distributions of the system uncertainties. The optimal dual controller is a function of the hyperstate, and hence it should be useful to have a representation of this quantity when constructing an approximately optimal dual controller. The hyperstate transition model is used in combination with a reinforcement learning algorithm for constructing a dual controller from stochastic simulations of a system model that includes models of the system uncertainties.

In the third paper, we suggest a simple reinforcement learning method that can be used to construct a decoupling matrix that allows MIMO control of a chiller process. Compared to the commonly used single-input single-output (SISO) structures, these controllers can decrease the variations in some system signals. This makes it possible to run the system at operating points closer to some constraints, which in turn can enable more energy-efficient operation.

Acknowledgements

Life is a complex uncertain dynamical system. Deciding what actions to take and when to take them is indeed an extremely challenging problem, for which not even the most brilliant scientists have been able to present a complete solution. However, as discussed in this thesis, there is at least one important factor that should be taken into account when it comes to control of such systems. The key concept to keep in mind is to try to find a balance between exploration and exploitation. My time at the Department of Automatic Control has amounted to a great deal of exploration. Although the rewards have varied over time, I have without question gained a lot of knowledge and experience that I hope to be able to exploit in the future. By empirical studies, I have concluded that research can be difficult and frustrating. However, I have also concluded that such situations of setbacks can be significantly ameliorated by being surrounded by good people. And the greatest feature of the department is that it is (and has been, during my whole time there) full of people who are not only very enthusiastic about and good at what they do, but, more importantly, also very friendly, open, helpful and nice in general. This fact has really enhanced my experience of working at the department.

I want to start by thanking my supervisor Bo Bernhardsson for sharing his enthusiasm for and knowledge about a wide range of topics through very pedagogical explanations, and for always being optimistic and helpful. I also want to thank my co-supervisor Anton Cervin for participating in and contributing to many research discussions and for punctuation improvements. Bo and Anton also made a great contribution by helping out with the proofreading of this thesis. Furthermore, I want to thank my other co-supervisor, Anders Rantzer, for always being open to discuss research and to contribute with his great ability to ask many interesting questions.

For the part of the thesis concerning chiller control, I had an excellent cooperation with the people at Carrier. I really appreciated to work with Bryan Eisenhower, who with great enthusiasm participated in regular meetings and contributed significantly by explaining all the control challenges for the process in question and by helping out with formulating suitable subproblems. Also Magdalena Atlevi and Kristian Tuszynski were very helpful for getting started with the chiller project.

Making a whole department run smoothly is also a highly complex control task that, nevertheless, is almost optimally solved through the cooperation between many excellent department controllers. One of these is Eva Westin, who, among many other contributions, has helped me to choose and execute actions in several difficult decision situations. Some others are Mika Nishimura, Cecilia Edelborg Christensen, and Monika Rasmusson, who always sort out all the important details, from making sure that different forms are filled out correctly and handling problematic conference registrations, to ensuring optimal coffee and fika management (and thus significantly enhancing the operational conditions of the department). Robustness against all possible kinds of technical problems, that never cease to appear, has always been ensured by Anders Nilsson, Anders Blomdell, Pontus Andersson, Leif Andersson, and Alexander Pisarevskiy.

Furthermore, I want to thank Emil Vladu for a great amount of moral support throughout the years, and for an uncountable number of contagious right-half-plane laughs. I also want to thank Sebastian Banert for good company, e.g. during lunches, for listening to a lot of random talk, and for patiently trying to correct all my German grammar mistakes. I am also grateful for having had, during several years, the opportunity to share an office with Johan Grönqvist, who has contributed with a lot of wisdom and enlightenment. I have appreciated all the discussions about everything from rocket science to different so-called "oj oj oj"-problems.

I also want to thank everyone else who I have met during my time at the department. Although I do not explicitly list everyone here, there are many more with whom I have had great interactions and whom I will never forget. Filling the breaks with random discussions, that often might appear to consist of pure nonsense, is not to be underestimated. These occasions can contribute with great rewards to the reinforcement learning problem of life. Something that, if not before, became apparent during the pandemic, when these became much rarer. Thus, this should also be taken into account when designing the control policy of life.

Lastly, I want to thank everyone outside of the department who also has supported me. Some of these are all the nice people I met during the years when I lived in a corridor at Delphi. I also want to thank my family, especially my parents, who always have supported me unconditionally, no matter what choices I have made.

Financial Support

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The work was also supported by the ELLIIT Strategic Research Area.

Contents

1. Introduction **9**
1.1 Automatic Control and Uncertainty 9
1.2 Contributions 11
2. Background **13**
2.1 Dual Control 13
2.2 Chiller Control 18
3. Publications **23**
4. Conclusions **25**
Bibliography **27**
Paper I. Dual Control of Linear Discrete-Time Systems with Time-Varying Parameters **29**
1 Introduction 30
2 Problem Formulation 31
3 Algorithm 31
4 Simulations 36
5 Conclusions 40
6 Acknowledgements 40
References 40
Paper II. Dual Control by RL Using Deep Hyperstate Transition Models **43**
1 Introduction 44
2 The Dual Control Problem 45
3 Hyperstate Transition Model 46
4 Example System 48
5 Hyperstate Transition Model – Results 50
6 Reinforcement Learning Algorithm 51
7 Control Results 54
8 Conclusion 56
9 Acknowledgements 58
References 58

Paper III. Model-Free Adaptive MIMO Control of a Chiller Process Using RL **61**
1 Introduction 62
2 Problem Formulation 74
3 Preliminaries 77
4 Method Description 82
5 Test Problem – Linearized Model 87
6 Results 91
7 Discussion 96
8 Acknowledgements 98
References 98

1

Introduction

1.1 Automatic Control and Uncertainty

In automatic control, we consider dynamical systems that can be manipulated by some input signal u, which is a function of time. In the general case, this signal is multidimensional, i.e., it consists of several components, so that $u = (u_1, u_2, \ldots, u_p)$, for some positive integer p. Our goal is to devise so-called controllers, which are algorithms that decide the value of the input u at all time points, such that the system behaves in a good way. To accomplish this, we first have to define what is meant by good behavior for the system. Second, we have to figure out how the input affects the system, so that it can be used to align the system as close as possible with the desired behavior.

A general system $\mathcal{S}$ can, except for the input u, also be affected by some external disturbance w. Moreover, it has some output y that can be measured and that might contain some measurement noise e. The system is characterized by its internal state x, which in general cannot be measured. All signals w, y, e, and x can be multidimensional. If the system model, as well as the input values $u(t)$ and disturbance values $w(t)$ for times $t \geq \tau$, are fully known, then it is sufficient to know the state value $x(\tau)$ at time $t = \tau$ in order to exactly predict future state values $x(t)$ for $t > \tau$. Thus, the state value $x(\tau)$ at a particular time τ can be thought of as containing all relevant information about the system that can affect its future behavior. Therefore, the system behavior can be characterized and described by its state value $x(t)$ at each time point t.

In conclusion, the control problem consists in constructing a controller $\mathcal{C}$ that generates a control signal u such that the state x of the controlled system $\mathcal{S}$ satisfies some predefined specification. This specification might involve a (possibly multidimensional) reference signal r that is used by the controller. The controller can also use the output signal y from the system. The setting is illustrated in Fig. 1.1.

The ideal case for a control designer would be if both the system model $\mathcal{S}$ and the disturbances w and e were fully known. Then, it would be possible to exactly predict what input u causes what state x. However, disturbances are typically not known. The second best case is then that we know the stochastic properties of the

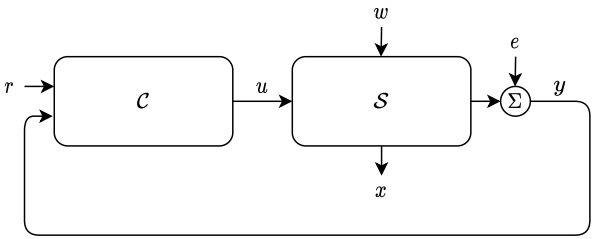


Figure 1.1 The control problem consists in designing a controller $\mathcal{C}$ that computes an input u for the system $\mathcal{S}$ with internal state x, under influence of an external disturbance w, using the measurement signal y with noise e, such that some specification, which might depend on the reference signal r, is satisfied.

signals, i.e., their probability distributions. In this case, we can determine the probability distribution of the resulting state for each given input. A famous example of a control method applicable to this case, if the system is linear and the disturbances and initial state value are normally distributed with known distributions, is linear-quadratic-Gaussian (LQG) control. This method designs a controller that minimizes the expected value of a quadratic cost that involves the state x and the input u.

In reality, the system we want to control cannot usually be perfectly described by a linear model. For some systems, a linear model might work for regulation around some operating point, although the system is nonlinear. However, there are also cases where a linear model does not adequately describe the behavior of the system even in the vicinity of a specific point. We might also want to control the system so that we move between distant operating points. In these cases, we have to look for methods beyond the well-known toolbox of linear control theory, most of which are much less developed and less general than the linear methods.

Except for nonlinearities, real-world systems usually also have different kinds of uncertainties. Even if the general structure of a system is known, the system often contains parameters that are not exactly known. The parameter values might be different for different realizations of the system, for different operating conditions, and might vary with time. The system $\mathcal{S}$ can therefore be regarded as a function $\mathcal{S}(\theta)$ of some stochastic parameter vector θ.

To find a controller that can handle cases where we have uncertain system parameters θ or a state x that is not directly measurable and uncertain, tools from the field of adaptive control can be used. The idea is then to let the control algorithm automatically change, depending on conditions such as current estimations of uncertain parameters and signals, and their uncertainties.

1.2 Contributions

The contents of this thesis can be divided into two main parts. In the first part, we consider the problem of adaptive control using a method called dual control. This method tries to balance the dual goals of actively collecting useful information about the system and using the available information to achieve good control. The problem with this method is that we have to keep track of the probability distributions of the uncertain parameters and variables of the system, as well as how these distributions and the system performance are affected by the choice of control signal (input) over a certain time horizon. In general, this problem has no analytical solution, and a straightforward numerical computation is impossible or intractable except for in some very simple special cases.

As for the dual control problem, our contributions consist in suggesting how to find approximate but computationally feasible solutions to this problem. We do this using tools from modern machine learning. One of the approaches is to use deep neural networks to approximate so-called value functions. These are functions that describe how good a certain state or combination of state and input is with regard to the goal of the dual control. This method is examined for linear systems with time-varying parameters. Another approach we explore is to construct a model that approximates how the probability distributions of the uncertainties change as a function of the input signal. Also for this model we make use of neural networks. This method is examined in a simple system with a nonlinear measurement function, for which linear methods and simple adaptive control supposedly are insufficient for achieving good control. The simplicity of the system makes it easy to illustrate that even a system of low complexity can be difficult to control if it cannot be described by a linear model, and that advanced control methods, in these cases, might be needed even for systems of low order.

The second part of the thesis considers control of a complex nonlinear system. The system in question is a so-called chiller, which is used to cool water, which in turn can be used to cool spaces such as buildings. For studying this system and investigating the effect of different control strategies on it, we have access to a complex nonlinear model, which can be used for simulations. In our setup, the system has only two inputs that should be chosen but has many different measurement signals that should be kept within certain bounds and that should be considered, in order to make sure that the control behavior is satisfactory. The model has in total 162 states, most of which usually cannot be directly measured in practice. This system can, at least sometimes, be adequately described locally using a linear model. We would like to apply some kind of adaptive control, to increase the control performance. However, it is important to keep the different variables within certain limits. The main challenge here is thus to implement some kind of adaptive control, without inducing too large variations in the constrained signals. In particular, we examine how combinations of measurement signals can be used as feedback to determine each of the two control signals. The control strategies that are employed

in currently manufactured processes usually determine each input independently, using only one measurement signal for each input. By using several measurements to determine each input, we show that control performance can be improved. Using such improved controllers can make it possible to run the process at operating points where the main control objective, i.e., to deliver the requested cooling capacity sufficiently fast, can be reached in a more energy-efficient way, making the process more economical and climate friendly.

2

Background

2.1 Dual Control

Exploration and Exploitation

When controlling a process of which we initially have incomplete information, the controller fulfils two different purposes. The first one is to perform experiments on the process, in order to yield more information about it that can be useful for determining future control actions. We refer to this as *exploration*. The second one is to perform the actual control, i.e., to try to align the process state with some given specification. We refer to this as *exploitation* of the available information. A controller that takes both of these purposes into account, and tries to balance them to achieve optimal control, is called a *dual controller*. This notion was introduced by Feldbaum [Feldbaum, 1960; Feldbaum, 1961], who also presented a general theoretical solution to the problem for the case of discrete-time systems where process changes can be modeled by a discrete Markov process. A good overview of some different dual control methods, as well as a short introduction to the concept of dual control, can be found in [Wittenmark, 1995]. This article, together with Feldbaum's papers, was the main inspiration for the remainder of this section. Some different examples of simple optimal and suboptimal dual controllers for different cases are described in, e.g., [Wittenmark, 1975] and [Åström and Helmersson, 1986]. More related references are presented in the papers in this thesis.

Feldbaum divides control settings into three different cases:

1. **Complete information:** Control of systems where all information about the process that can be known is known. This encompasses the process model, information about process disturbances w (see Fig. 1.1), the state x of the process, and the control objective.

2. **Incomplete information with passive accumulation of information:** Control of systems where some information that could be known is not known and where the controller does not have any strategy for exploration, so that information is only collected passively.

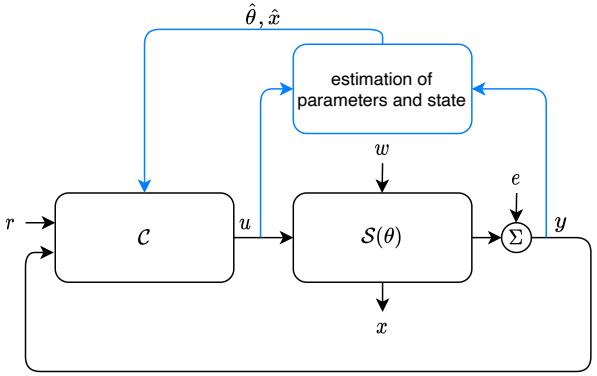


Figure 2.1 In *classical adaptive control*, the controller $\mathcal{C}$ is modified based on estimates of uncertain quantities, such as e.g. system parameters θ or a not directly measurable state x. The estimates of the uncertainties are based on all previous and current values of the control signal u and measurement signal y from the system $\mathcal{S}(\theta)$. The control aim can involve a reference signal r, while the system can be affected by disturbances w, and the output y can be corrupted by noise e.

3. **Incomplete information with active accumulation of information:** Control of systems where some information that could be known is not known, and where the controller has an explicit exploration policy to find some of this information.

Most variants of adaptive control, and the ones that are most commonly used, fall into the second of these categories. Dual control, on the other hand, corresponds to the third category.

Classical Adaptive Control

Classical adaptive controllers are usually composed as illustrated in Fig. 2.1. Using the control signal u and the measurement signal y, the state x of the system as well as unknown system parameters θ can be estimated. These estimates can then be used by the controller to adapt future control actions. One common approach is so-called *certainty equivalence* control. In this method, a controller is designed assuming that complete process information is available. The process might, e.g., be parameterized with a parameter vector θ, and the controller is then designed using the process model, assuming that θ is known. This controller is then used with θ replaced by some estimate $\hat{\theta}$ to control the real process. This approach often works well if the estimate $\hat{\theta}$ is sufficiently accurate. However, if it is not, the closed-loop behavior can differ radically from the intended one.

To take into account the uncertainty of the process, one slightly more advanced variant of adaptive control that can be used is so-called *cautious control*. In this ap-

proach, the control signal at a particular time step is determined by minimizing the expected value of some cost for the process variables at the next time step, given the current estimated mean and covariance of the parameter θ. The effect of this method is usually that smaller control signals are applied when the uncertainty is large, hence the name. Although this in many situations is a desirable property, it might also cause problems. Keeping the control signal small means that we maintain a small degree of exploration, i.e., we do not collect a lot of new useful information about the process. This tends to lead to that the process uncertainty increases even more, which in turn decreases the magnitude of the control signal further. Thus, we have a vicious circle that can lead to the result that the control signal eventually remains zero, while the uncertainty remains large. This is called the *turn-off phenomenon*. [Åström and Wittenmark, 2011]

Optimal Dual Control

In optimal dual control, the exploration-exploitation trade-off is taken into account by trying to minimize some loss function over a given time horizon. Let x_k denote the value of the state x at time step k, and analogously for the other signals. Furthermore, let $\mathcal{Y}_k = \{y_k, y_{k-1}, \ldots, u_{k-1}, u_{k-2}, \ldots\}$ be the set containing the last measurement signal as well as all previous inputs and outputs. Using a time horizon of T steps, the goal can then be formulated as that we want to select the control signal u_k such that we minimize a loss function

$$J_k = \mathbb{E}\left\{ \sum_{j=k}^{k+T-1} c_j(x_{j+1}, u_j) \,\middle|\, \mathcal{Y}_k \right\}, \tag{2.1}$$

where $\mathbb{E}\{\cdot\}$ denotes mathematical expectation and c_j are some cost functions.

Cautious control corresponds to choosing the time horizon as $T = 1$. In this case, the optimal solution does not encompass any explicit exploration, since the information gained from exploration would be useful only in future time steps, which are not taken into account by the cost. Thus, to promote control strategies that actively collect information about the system that can improve future control, the control horizon must be chosen as $T \geq 2$. In that case, a larger step cost in early time steps due to exploration can be compensated by smaller step costs in the later time steps due to the gained information.

The optimal dual controller, i.e., the controller that minimizes (2.1), is given as the solution to a functional equation called the *Bellman equation*. Before presenting this equation, we introduce two new concepts. The first one is the *hyperstate*, which is the joint probability distribution of the system state x_k and system parameters θ_k at a time point k, given the input and output history $\mathcal{Y}_k$. The hyperstate is denoted by

$$\xi_k = \mathbb{P}(x_k, \theta_k \mid \mathcal{Y}_k).$$

The second concept is the value function

$$V_\tau(\xi_k, k) = \min_{u_k,\ldots,u_{k+\tau-1}} \mathbb{E}\left\{ \sum_{j=k}^{k+\tau-1} c_j(x_{j+1}, u_j) \,\middle|\, \mathcal{Y}_k \right\}.$$

This function is equal to the minimizing value of the loss (2.1) when $\tau = T$. Using dynamic programming, the value functions can now be computed recursively by the Bellman equation

$$V_\tau(\xi_k, k) = \min_{u_k} \mathbb{E}\left\{ c_k(x_{k+1}, u_k) + V_{\tau-1}(\xi_{k+1}, k+1) \mid \mathcal{Y}_k \right\}, \tag{2.2}$$

starting with $V_0(\xi_k, k) = 0$ and then solving for $V_1, V_2, \ldots, V_T$. The optimal input at time k is then given by

$$u_k^* = \arg\min_{u_k} \mathbb{E}\left\{ c_k(x_{k+1}, u_k) + V_{T-1}(\xi_{k+1}, k+1) \mid \mathcal{Y}_k \right\}.$$

It can be shown that the dependence of this optimal control signal on the signal history $\mathcal{Y}_k$ is covered by the information contained in the hyperstate ξ_k [Bertsekas, 1987]. Therefore, we obtain an optimal control policy π_T that minimizes the loss (2.1) with time horizon T in the form

$$u_k^* = \pi_T(\xi_k).$$

Hence, the optimal dual controller is a policy that determines the control signal u_k at time k as a function of the hyperstate ξ_k at the same time point. The controller structure is therefore as illustrated in Fig. 2.2. For this control structure, we need an estimator that estimates the hyperstate ξ_k at each time point given the input and output history of the system as well as prior knowledge of the initial uncertainties. A simple instance of this would be a case where the joint probability distribution of the state x and parameter vector θ could be assumed to be a normal distribution. In this case, the hyperstate ξ_k would consist of the mean and covariance of this distribution. This could be compared to the classical adaptive control structure in Fig. 2.1, where the estimator only provides means of these quantities but no other information about their probability distributions to the controller.

Suboptimal and Approximately Optimal Dual Control

Although solving the Bellman equation (2.2) would yield the optimal dual controller that minimizes the loss (2.1), this equation is unfortunately either impossible or intractable to solve in practice, except for some very simple cases. This is due to the nesting of the mathematical expectation and minimization that is needed for the recursive solution.

One way to circumvent this is to use some kind of suboptimal dual control instead of optimal dual control. One simple method for incorporating some exploration into the control mechanism is, e.g., to use the cautious controller plus some

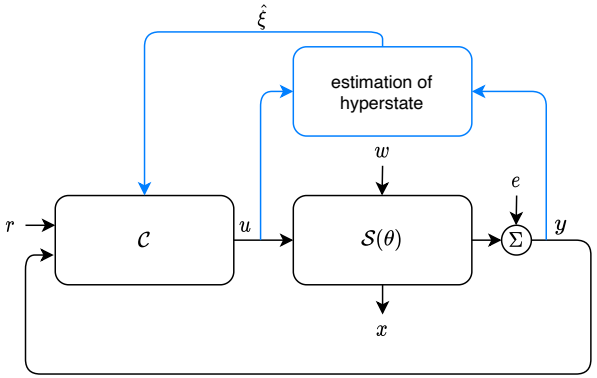


Figure 2.2 In *optimal dual control*, the controller $\mathcal{C}$ applies a policy based on an estimation of the hyperstate ξ, which is the joint probability distribution of uncertain quantities in the system, such as e.g. system parameters θ or a not directly measurable state x. The estimate of the hyperstate is based on all previous and current values of the control signal u and measurement signal y from the system $\mathcal{S}(\theta)$. The control aim can involve a reference signal r, while the system can be affected by disturbances w, and the output y can be corrupted by noise e.

added perturbation, such as white noise of some amplitude. A perturbation added to an adaptive controller could also be a function of the current uncertainty, so that its amplitude is increased when the uncertainty is large, in order to increase the amount of exploration in that case.

A more advanced approach is to try to construct an approximately optimal dual controller by approximately solving the Bellman equation. An exact solution would require us to compute the value of the value function V_τ for every possible hyperstate ξ in each recursion. Since there are an infinite number of possible hyperstate values, this is of course not possible. We could, however, evaluate the value function for a certain set of hyperstate values and then use some kind of function interpolation method that can yield approximate function values for any value of the hyperstate. Since the value function in general is a nonlinear function even if the system that is studied is linear, we need to choose a function interpolation method that can represent nonlinear functions with a complex structure. One such method, which has had great success in several fields in recent years, is deep neural networks. In Paper I of this thesis, we examine the possibility of using neural networks in this way, i.e., to approximate the value functions used for solving the Bellman equation, for a simple example system.

Another possible way to achieve approximately optimal dual control could be to apply reinforcement learning (RL) methods to a stochastic simulation model of the system, containing probability distributions for all the system uncertainties. Such a method could then be used to find a policy that gives low values of the loss function

(2.1). The idea here would be to use reinforcement learning with the hyperstate as the reinforcement learning state, instead of an ordinary system state. To use this method, we need a way to effectively and efficiently represent and estimate the hyperstate at each time point, given information about the input and output signals of the system. This is needed both when applying the resulting policy to a real system, and also for being able to run many iterations of the RL algorithm on the simulated model within a limited amount of time. In Paper II, we suggest a method for constructing a so-called hyperstate transition model (HTM) that can be used for this purpose. We assume that the hyperstate can be represented as a mixture model and train a neural network using results from simulations with the stochastic model to return the parameters defining the new hyperstate given the parameters of the previous hyperstate as well as the most recent input and output values. The approach of using this HTM in combination with reinforcement learning on a simulated system to construct a dual control policy is tested on a simple but nontrivial example system, where the exploration induced by a dual controller is needed in order to achieve good control performance.

2.2 Chiller Control

Motivation

According to the International Energy Agency (IEA), global energy consumption due to space cooling, i.e. due to the use of air conditioning systems, has more than tripled since year 1990. In year 2020, the total energy consumption due to such systems was about 1 885 TWh, corresponding to about 8% of the global electricity consumption. [IEA, 2022; Statista, 2022]

In order to bound the global mean temperature in accordance with the Paris agreement, the global emissions of greenhouse gases must be drastically reduced in the near future. One important contribution to this would be to make equipment that consumes a lot of energy more energy efficient. Since air conditioning systems is an important category of such systems, which is becoming even more important, making these more energy efficient would lead to a significant contribution to decreasing global energy consumption. Another motivation for doing this is that the load on electricity grids that can get very large at peak hours, and especially during heat waves, would be decreased. Moreover, at the time of writing this (September 2022), an energy crisis is emerging in Europe, which gives additional strong economical incentives for increasing energy efficiency wherever possible. Since heat pumps are constructed as inverted air conditioning systems, more efficient control methods for cooling systems could lead to energy savings also for heat pumps.

Process Description

The purpose of a chiller is to cool down some gas or liquid that is used for cooling down a space, e.g., a building. The medium used for cooling is called *coolant*

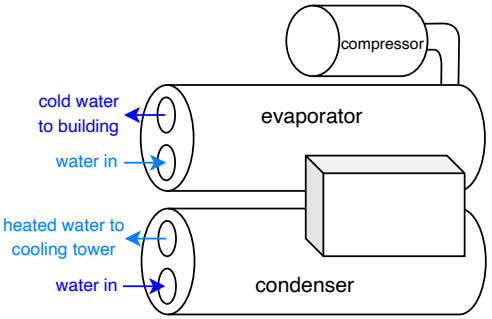


Figure 2.3 The main components of the chiller process.

and circulates between the building and the chiller. The task of the machine is to decrease the temperature of the coolant, or, in other words, extract heat from it, so that the coolant in turn can be used to continuously absorb heat from the space to be cooled. The example system in this work uses water as a coolant.

The chiller has three main components: an evaporator, a condenser, and a compressor, as shown in Fig. 2.3. The water that circulates through the building to be cooled is passed through the evaporator, where it is cooled down. By means of another liquid, the *refrigerant*, for which the pressure is varied, the heat from the evaporator water can be transferred to water in a separate circulation, through the condenser. The water that goes through the condenser is circulated to a cooling tower, where it is cooled down, e.g., using outdoor air.

In the evaporator, the water that circulates through the building is cooled using a liquid called refrigerant. This is a liquid that has chemical properties that are favorable for the application in question. In the choice of refrigerant, the environmental impact must also be taken into account, often forced by governmental regulations that tend to become stricter over time, which could make controlling the process more challenging. The water passes through the evaporator in pipes that are surrounded by refrigerant. This is illustrated in Fig. 2.4. The refrigerant absorbs heat by transforming from liquid to gas.

In the condenser, shown in Fig. 2.5, the heat absorbed by the refrigerant is transferred to the condenser water. When the refrigerant releases heat, it condenses, i.e., transforms from gas to liquid. Just like in the evaporator, the water is passed through the condenser in pipes that are surrounded by refrigerant.

To achieve the right phase transitions for absorbing and emitting heat in the evaporator and condenser, the refrigerant has to have a specific pressure in each case. The refrigerant thus passes through a cycle where it undergoes changes in pressure, when it passes between the evaporator and the condenser.

Before describing the refrigerant cycle, we introduce a measure of thermody-

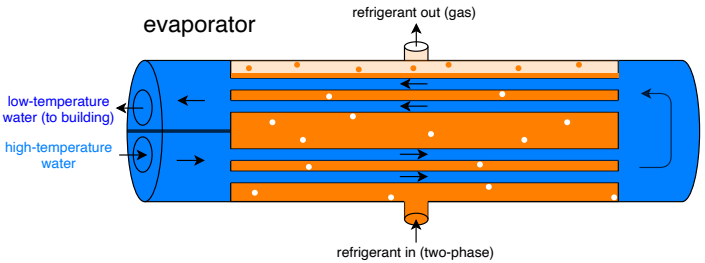


Figure 2.4 In the *evaporator*, water (blue) is cooled down by transferring heat to the refrigerant (orange). The refrigerant enters as two-phase and leaves as a gas.

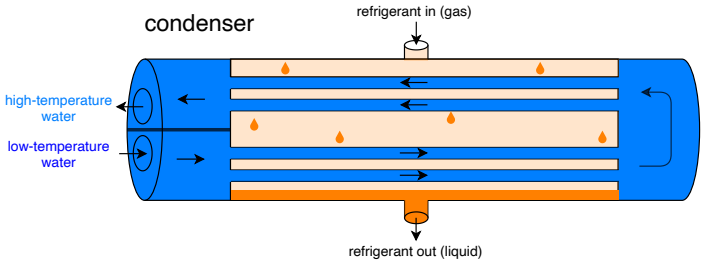


Figure 2.5 In the *condenser*, heat is emitted to the circulating water (blue). The refrigerant (orange) enters as a gas, emits heat to the water by condensation, and leaves as a liquid.

namic potential, called *enthalpy*. The enthalpy H of a fluid is defined as

$$H = U + pV,$$

where U is the inner energy, p the pressure, and V the volume. [IUPAC, 1997] Under constant pressure, it holds that

$$\mathrm{d}H = \mathrm{d}Q \overset{*}{=} C_p \mathrm{d}T, \quad * \text{ if no phase transitions},$$

where Q is absorbed heat, C_p is a constant (heat capacity at constant pressure), and the second equality (*) is valid only under the assumption that there are no phase transitions. Thus, absorption of a certain quantity of heat leads to a change of the enthalpy by the same amount. Furthermore, if there is no phase change when the heat is absorbed, the change in temperature is proportional to the amount of absorbed heat.

Using the notion of enthalpy and its properties, we can now describe the refrigerant cycle. The cycle can be illustrated using a pressure–enthalpy (ph) diagram, as

shown in Fig. 2.6. On the vertical axis, we have the pressure p, and on the horizontal axis, we have the specific enthalpy h, defined as enthalpy per unit mass of the fluid. The blue curve divides the plane into three parts, corresponding to the fluid being a liquid, a gas, or two-phase, i.e., a mix of liquid and gas. The black curve shows the refrigerant cycle, and the orange lines are curves along which the temperature is constant.

Starting at point 1 in the figure, the refrigerant leaves the evaporator as a gas. The pressure of this gas is increased by the compressor, and led to the condenser. In the compression, there is also an increase in enthalpy, corresponding to the minimal amount of energy required for the compressor to achieve the pressure increase. Between points 1 and 2 in the figure, the compressed refrigerant increases in temperature, which can be seen by the fact that several temperature level curves are crossed. Between points 2 and 3, the warm gas enters the condenser, where it is cooled down by the condenser water, so that it condenses. At point 3, the fluid has completely turned into a liquid. From here to point 4, the liquid runs through an expansion valve, so that the pressure of it decreases. The pressure drop leads to some of the refrigerant turning into gas again, i.e., we return to the two-phase region. Since the refrigerant pressure continues to decrease in the two-phase region, the temperature is decreased as well, as seen by the temperature level curves. Between points 4 and 1, the cooled-down two-phase refrigerant takes up heat from the evaporator water, until it has evaporated, so that we have a gas that again is sent to the compressor.

Goal

In the problem setting that we study in this thesis, we have a chiller process with two actuators: the compressor and the expansion valve. The overall goal is to construct a controller that uses these two actuators to achieve the control goals within the given constraints of the process in an as energy efficient way as possible. The most important goal of the controller, given that all constraints are satisfied, is to make sure that the water leaving the evaporator achieves the specified temperature sufficiently fast. However, there are in general many ways to achieve this, of which some are more energy efficient than others.

In the controller structures that are usually implemented in real chiller processes today, each of the two actuators are set independently to values determined using one measurement signal for each actuator. In this thesis, we examine whether it is possible to improve control performance by determining the actuator values jointly, based on several measurement signals. To be able to achieve efficient control, the system might have to operate in states where some variables are close to boundaries that must not be exceeded. To do this, it is important that the deviations of these variables from their nominal values caused by system disturbances and reference changes are as small as possible. In Paper III, we examine the subproblem of constructing controllers to set the two actuator values that keep deviations for some

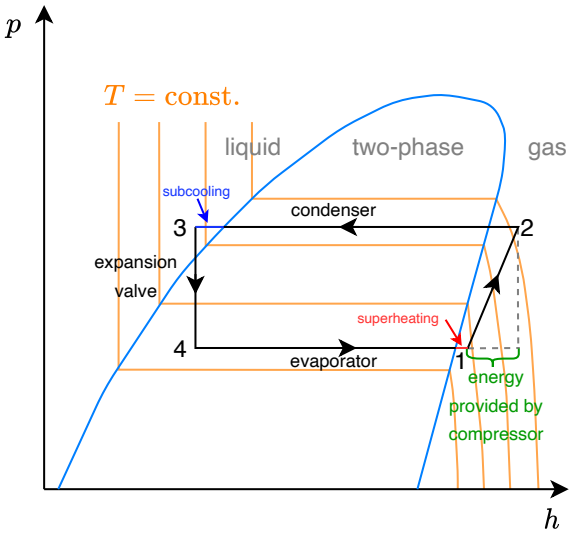


Figure 2.6 Pressure-enthalpy (ph) diagram with the refrigerant cycle (black), the phase-transition curve for the refrigerant (blue), and some level-curves for the temperature (orange). The pressure is varied by using a compressor and an expansion valve. This makes it possible to absorb heat in the evaporator, and emit heat in the condenser.

relevant measurement signals small. This is done by applying a simple reinforcement learning method to simulations of the system, in order to learn a decoupling matrix that combines outputs of two PI controllers to determine the two control signals. The approach of jointly controlling the two actuators is compared with the nominal case of decentralized control.

3

Publications

This licentiate thesis is based on the following three publications. The contributions of each author are described below.

Paper I

Rosdahl, C. and Bernhardsson, B. (2020). "Dual Control of Linear Discrete-Time Systems with Time-Varying Parameters". *2020 International Conference on Control, Automation and Diagnosis (ICCAD)* (pp. 1-6). IEEE.

The problem formulation and method were developed in discussions between the authors. C. Rosdahl derived the mathematical expressions for computing the value function for a general discrete-time linear system with time-varying parameters, as a generalization of the method used in another paper for the case of an integrator with unknown gain. C. Rosdahl constructed, implemented and ran simulations with the algorithm for control law computation based on approximating the value function with a neural network. B. Bernhardsson wrote some background and about some related work in the manuscript and helped with proofreading the whole manuscript. C. Rosdahl wrote the remaining parts of the manuscript.

Paper II

Rosdahl, C., Cervin, A. and Bernhardsson, B. (2022). "Dual Control by Reinforcement Learning Using Deep Hyperstate Transition Models". *IFAC-PapersOnLine*, 55(12), 395-401.

The idea of using a neural network model for the transition of some representation of the hyperstate, to be used in combination with dual control, was suggested by B. Bernhardsson. The details of the hyperstate transition model construction algorithm were formulated by C. Rosdahl, who also implemented and tested the algorithm. The reinforcement learning algorithm for constructing the action–value function

using the hyperstate transition model was formulated, implemented, and tested by C. Rosdahl. The test examples and the general structure of the overall method were decided by regular discussions between all authors. B. Bernhardsson contributed to the introduction part of the manuscript by describing some background and giving context to the method. C. Rosdahl wrote most of the remaining parts of the manuscript. A. Cervin extended and improved the text in several sections. A. Cervin and B. Bernhardsson also helped with structuring and proofreading the manuscript.

Paper III

Rosdahl, C., Bernhardsson B. and Eisenhower B. (2022). "Model-Free Adaptive MIMO Control of a Chiller Process Using Reinforcement Learning". *Manuscript prepared for journal submission.*

The overall problem formulation was given by B. Eisenhower. All of the authors contributed significantly to the development of the method, participating in regular discussions during its development. The details of the reinforcement learning algorithm, including the cost used for the test problem, were formulated and implemented by C. Rosdahl. The introduction section of the manuscript, with general background information on control of vapor compression systems, was written by B. Eisenhower. The preliminaries about decoupling were written by B. Bernhardsson. The remaining parts of the manuscript were written by C. Rosdahl.

4

Conclusions

The main conclusion of this thesis is that the combination of classical methods from control theory with modern methods from the field of machine learning has the potential to pave the way for new interesting algorithms that can solve problems for which simple classical control methods are insufficient. While machine learning methods for some applications tend to yield impressive results, these usually require huge amounts of data in order to work well. Automatic control can, on the other hand, perform relatively well given very little information if we have a sufficiently good model that describes the structure of the system to be controlled. Thus, control theory let us utilize known structures in an efficient way, while machine learning contributes with a great amount of flexibility and adaptability. It is therefore a reasonable idea that combining tools from both of these fields should be able to give us the best of both worlds.

In Paper I, we use classical control theoretical results for how to achieve optimal dual control, i.e., for how to find a controller that balances exploration and exploitation in order to minimize some control cost over a given horizon. Since these results require us to compute and store values of highly nonlinear multidimensional functions, methods for approximately doing this is an efficient way are needed. To this end, we utilize artificial neural networks, which are very flexible structures that have the potential to represent complex nonlinear functions. Simulations with some simple examples suggest that this method can indeed be useful. Future work would be to apply the same or similar methods to more complex systems and to further examine what structures for neural networks are best adapted to this application.

In Paper II, we again return to the dual control problem. In this case, we do not try to solve the Bellman equation (the equation that yields the optimal dual controller) explicitly, but instead try to construct an estimator that represents the hyperstate, i.e. the probability distribution of the system's uncertainties, at each time step. Also for this nonlinear function, we make use of a neural network. Classical control theory suggests that the hyperstate is the important quantity for determining the optimal dual controller. Therefore, we try to use this estimator in combination with a simple reinforcement learning method applied to a simulation of the system, where the uncertainties are included in the model, to find a dual controller

that can be used for controlling the system. Tests on an example system show that the method succeeds in constructing a controller with dual features, i.e. a controller that applies probing actions when this is advantageous for minimizing a given control cost over some time horizon. Future work would involve applying the method to more complex systems and settings. Also exploring different structures for the neural network in the hyperstate transition model as well as different kinds of reinforcement learning methods to use with this model, would be interesting tracks for future research.

In Paper III, we use classical PI controllers and classical control theory results for how to perform multivariate control by using a decoupling structure. However, instead of assuming that we have a perfect model that can be used to compute a fixed static decoupling, we utilize a simple reinforcement learning algorithm that has the ability to adjust the decoupling adaptively. By simulation studies, we see that the method is able to find a good decoupling even starting from a completely decentralized controller structure. In practice, this method could be used to adapt controllers to different versions of the process in question, or to changes in a specific process that occur over time. Thus, it is yet again observed that classical control and machine learning can be combined into interesting new algorithms that can be useful for solving practical problems. Future work should involve testing the method with the complex nonlinear model on more complex settings, e.g. involving more different measurement signals. Eventually, the method should be tested in combination with a high-level optimizer, first in simulations and then on real processes.

In this thesis, we have given a few demonstrations of algorithms that combine control theory and machine learning, with the aim of constructing adaptive controllers that can achieve good control over some time-horizon. The fast development of machine learning methods in recent years offers great potential for further investigating such approaches. Hopefully, this thesis can contribute to encouraging more researchers to continue exploring in this direction.

Bibliography

Åström, K. J. and A. Helmersson (1986). "Dual control of an integrator with unknown gain". *Comput. Math. with Appl.* **12**:6, Part A, pp. 653–662. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(86)90052-0.

Åström, K. J. and B. Wittenmark (2011). *Computer-controlled systems: theory and design*. 3rd ed. Dover Publications Inc., p. 431.

Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Inc., USA.

Feldbaum, A. A. (1960). "Dual control theory I-II". *Autom. Remote Control* **21**, pp. 874–880, 1033–1039.

Feldbaum, A. A. (1961). "Dual control theory III-IV". *Autom. Remote Control* **22**, pp. 1–12, 109–121.

IEA (2022). *Cooling*. https://www.iea.org/reports/cooling.

IUPAC (1997). *Compendium of Chemical Terminology, 2nd ed. (the "Gold Book")*. Blackwell Scientific Publications, Oxford. Online version (2019-) created by S. J. Chalk. ISBN 0-9678550-9-8. https://doi.org/10.1351/goldbook.

Statista (2022). *Net electricity consumption worldwide in select years from 1980 to 2019*. https://www.statista.com/statistics/280704/world-power-consumption/.

Wittenmark, B. (1975). "An Active Suboptimal Dual Controller for Systems with Stochastic Parameters". *Automatic Control Theory and Application* **3**, pp. 13–19.

Wittenmark, B. (1995). "Adaptive Dual Control Methods: An Overview". *IFAC Proceedings Volumes* **28**:13, pp. 67–72. ISSN: 14746670. DOI: 10.1016/s1474-6670(17)45327-4.

Paper I

Dual Control of Linear Discrete-Time Systems with Time-Varying Parameters

Christian Rosdahl **Bo Bernhardsson**

Abstract

We describe how the optimal dual controller for a discrete-time linear system can be found by approximately solving the corresponding Bellman equation using a neural network to represent the value function. We illustrate the method on an example with time-varying dynamics, where the new method is shown to give improved performance.

Originally published in the 2020 International Conference on Control, Automation and Diagnosis (ICCAD), Paris, France, October 7–9, 2020. Reprinted with permission.

1. Introduction

Adaptive control of systems with unknown dynamics is traditionally addressed by first designing a controller under the assumption that the system parameters are known, and then using the designed controller with estimations of the parameters in place of the real parameters. This approach is called *certainty equivalence* control. If the estimates are poor, the resulting control could however differ drastically from the aims of the control design. Neither is it certain that the system will operate in a manner such that the parameter estimates will improve with time.

By choosing the control signal better, it is possible to excite the system in order to deliberately improve the parameter estimates, so-called *probing*. This gives rise to the *exploration-exploitation trade-off*, which aims to select the probing in order to optimize long-term control, albeit at the expense of short-term control. This is the essence of what, due to the dual aims of estimating and controlling, is denoted *dual control*. This regime was introduced by Feldbaum [Feldbaum, 1960; Feldbaum, 1961], who showed that the problem could, in principle, be solved by solving a certain functional equation – the Bellman equation.

The remaining issue for the dual control is that the Bellman equation is intractable to solve exactly, except for some very simple cases. This lead to a range of approximate and suboptimal dual control methods some years ago, leading to some partial progress, see e.g. [Tse et al., 1973; Tse and Bar-Shalom, 1973], [Wittenmark, 1975] and [Lindoff et al., 1998]. An overview of some of the methods is given in [Wittenmark, 1995]. Some more recent approaches are described in [Svensson, 2016] and [Bayard and Schumitzky, 2010]. In recent years, there has been a renewed interest in the area due to the rapid progress in reinforcement learning, where general model-free algorithms have been developed, which perform well on many different problems, though often requiring a long learning time before successful control is achieved, see [Sutton and Barto, 2018]. It is therefore of interest to try to combine the classical control methods with modern machine learning methods, as proposed in e.g. [Recht, 2019] and [Klenske and Hennig, 2016], in order to utilize the advantages of both of the approaches.

In this paper we describe how the model-based traditional approach can be renewed by introducing recent machine learning based methods to find nonlinear function approximators used for solving the Bellman equation.

We will describe a generalization of the algorithm from [Åström and Helmersson, 1986] that applies to a general discrete-time system with unknown normally distributed parameters which might vary in time. The algorithm can be used with any suitable methods for evaluation point selection and interpolation of the value function. One possible choice, consisting of random uniform sampling and interpolation by neural networks, is explored by application on two simple examples.

2. Problem Formulation

We will consider linear systems of the form

$$y_k = \sum_{i=1}^{n} a_k^i y_{k-i} + \sum_{i=1}^{m} b_k^i u_{k-i} + e_k, \tag{1}$$

where $\{e_k\}$ denotes independent normally distributed noise with $\mathbb{E}\{e_k\} = 0$ and $\mathbb{E}\{e_k^2\} = \sigma^2$. We will for simplicity assume that the value of the constant parameter σ is known. Note that we have allowed the a and b coefficients to be time-varying. This enables the study of control policies for slowly evolving dynamical systems.

The system can be rewritten as

$$\begin{aligned} y_{k+1} &= \theta_{k+1}^{\mathrm{T}} \phi_k + e_{k+1}, \text{ where} \\ \theta_k &= \begin{bmatrix} b_k^1 & a_k^1 & \dots & a_k^n & b_k^2 & \dots & b_k^m \end{bmatrix}^{\mathrm{T}}, \\ \phi_k &= \begin{bmatrix} u_k & x_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \\ x_k &= \begin{bmatrix} y_k \dots y_{k-n+1} & u_{k-1} \dots u_{k-m+1} \end{bmatrix}^{\mathrm{T}}. \end{aligned} \tag{2}$$

The vector ϕ_k contains the input u_k to be chosen at time k, as well as the previous inputs and outputs, collected in the vector x_k.

The system parameters in θ_k are not known, but are assumed to have a prior distribution $\theta_0 \sim \mathcal{N}(\hat{\theta}_0, P_0)$. Furthermore, the parameters are assumed to evolve according to

$$\theta_{k+1} = A\theta_k + v_k, \tag{3}$$

for some known matrix A, where $v_k \sim \mathcal{N}(0, R_v)$ and $\mathbb{E}[v_k v_j^{\mathrm{T}}] = 0$ for all $j \neq k$.

The goal at time k is to determine control laws $\{u_j(\mathcal{Y}_j)\}_{j=k}^{k+T-1}$ that minimize the loss

$$J = \mathbb{E}\left\{ \sum_{j=k+1}^{k+T} y_j^2 \,\middle|\, \mathcal{Y}_k \right\}, \tag{4}$$

where $\mathcal{Y}_k = \{y_k, y_{k-1}, \dots, u_{k-1}, u_{k-2}, \dots\}$ is the set of observed outputs and previous inputs.

3. Algorithm

3.1 Hyperstate

In the solution of the optimal control problem, we use the *hyperstate* ξ_k consisting of the process state and the conditional distribution of the parameter vector θ_{k+1} given $\mathcal{Y}_k$.

The conditional distribution of θ_{k+1} at any given time is Gaussian and is therefore fully described by the mean and covariance

$$\hat{\theta}_{k+1|k} = \mathbb{E}\{\theta_{k+1} \mid \mathcal{Y}_k\}, \tag{5}$$

$$P_{k+1|k} = \mathbb{E}\{(\theta_{k+1} - \hat{\theta}_{k+1|k})(\theta_{k+1} - \hat{\theta}_{k+1|k})^{\mathrm{T}} | \mathcal{Y}_k\}. \tag{6}$$

The hyperstate at time k contains a sufficient statistics for the future statistical optimization problem and is given by

$$\xi_k = (x_k, \hat{\theta}_{k+1|k}, P_{k+1|k}). \tag{7}$$

The conditional mean and covariance of the parameter vector can be recursively computed by applying a Kalman filter [Åström and Wittenmark, 2011] to the system defined by (2) and (3), with the initialization $\theta_0 \sim \mathcal{N}(\hat{\theta}_0, P_0)$. The mean value $\hat{\theta}_{k+1|k}$ and estimation error covariance $P_{k+1|k}$ for the parameter vector θ_{k+1} given $\mathcal{Y}_k$ can then be computed as

$$\hat{\theta}_{k+1|k} = A\hat{\theta}_{k|k-1} + s_{k-1}^{-2} A P_{k|k-1} \phi_{k-1}^{\mathrm{T}} \varepsilon_k, \tag{8}$$

$$P_{k+1|k} = A P_{k|k-1} A^{\mathrm{T}} + R_v - s_{k-1}^{-2} A P_{k|k-1} \phi_{k-1}^{\mathrm{T}} \phi_{k-1} P_{k|k-1} A^{\mathrm{T}}, \tag{9}$$

where $\varepsilon_k := y_k - \phi_{k-1}^{\mathrm{T}} \hat{\theta}_{k|k-1}$ and

$$s_k := \sqrt{\phi_k^{\mathrm{T}} P_{k+1|k} \phi_k + \sigma^2}. \tag{10}$$

3.2 The Bellman Equation

The problem we want to solve is to choose $\{u_j\}_{j=k}^{k+T-1}$ such that the loss function (4) is minimized. To accomplish this, we make use of the value function

$$V_\tau(\xi_k, k) = \min_{u_k, \ldots, u_{k+\tau-1}} \mathbb{E}\left\{ \sum_{j=k+1}^{k+\tau} y_j^2 \,\middle|\, \mathcal{Y}_k \right\}, \tag{11}$$

which is equal to the loss (4) when k is the current time and $\tau = T$. Using the principle of optimality, the value function gives the Bellman equation

$$V_\tau(\xi_k, k) = \min_{u_k} \mathbb{E}\left\{ y_{k+1}^2 + V_{\tau-1}(\xi_{k+1}, k+1) \mid \mathcal{Y}_k \right\}. \tag{12}$$

The linearity property of the expected value allows for computing the expectation of each term separately. The first term can, by insertion of the system dynamics (2), be computed as

$$\mathbb{E}\{y_{k+1}^2 | \mathcal{Y}_k\} = (\phi_k^{\mathrm{T}} \hat{\theta}_{k+1|k})^2 + \phi_k^{\mathrm{T}} P_{k+1|k} \phi_k + \sigma^2. \tag{13}$$

To compute the second part of the expectation in (12), we need to express the hyperstate $\xi_{k+1} = (x_{k+1}, \hat{\theta}_{k+2|k+1}, P_{k+2|k+1})$ in terms of the current hyperstate ξ_k and the

input to be decided, u_k. Given ξ_k and u_k, all components of x_{k+1} are known except for the stochastic variable

$$y_{k+1}(\xi_k, u_k) = \phi_k^{\mathrm{T}} \theta_{k+1} + e_{k+1}, \tag{14}$$

where ϕ_k contains known values, whereas θ_{k+1} and e_{t+1} are independent normally distributed variables. After adding the two independent Gaussian variables, we can rewrite this as

$$y_{k+1}(\xi_k, u_k; X) = \phi_k^{\mathrm{T}} \hat{\theta}_{k+1|k} + s_k X, \tag{15}$$

where $X \sim \mathcal{N}(0,1)$ is the only stochastic component in (15) and s_k is defined in (10).

The variable $\hat{\theta}_{k+2|k+1}$ is according to (8) and by insertion of (15) given by

$$\hat{\theta}_{k+2|k+1}(\xi_k, u_k; X) = A\hat{\theta}_{k+1|k} + s_k^{-1} A P_{k+1|k} \phi_k X, \tag{16}$$

where X is the same random variable as in (15). The final hyperstate variable $P_{k+2|k+1}$ can be expressed in known variables according to (9), as

$$P_{k+2|k+1}(\xi_k, u_k) = A P_{k+1|k} A^{\mathrm{T}} + R_v - s_k^{-2} A P_{k+1|k} \phi_k^{\mathrm{T}} \phi_k P_{k+1|k} A^{\mathrm{T}}. \tag{17}$$

Note that $P_{k+2|k+1}$ does not depend on X, but is deterministic given ξ_k and u_k.

Using these calculations we can now express the Bellman equation as

$$\begin{aligned} V_\tau(\xi_k, k) = \min_{u_k} \Big\{ &(\phi_k^{\mathrm{T}} \hat{\theta}_{k+1|k})^2 + \phi_k^{\mathrm{T}} P_{k+1|k} \phi_k + \sigma^2 \\ &+ \int_{-\infty}^{\infty} \varphi(X) V_{\tau-1}(\xi_{k+1}(\xi_k, u_k; X), k+1)\, \mathrm{d}X \Big\}, \end{aligned} \tag{18}$$

where $\varphi(X) = (1/\sqrt{2\pi}) e^{-X^2/2}$ is the probability density function for the standard normal distribution, and $\xi_{k+1}(\xi_k, u_k; X)$ is given by (15)-(17).

Using the fact that $V_0 \equiv 0$, the optimal control u_k^* at time k can now be determined by recursive solution of (18). By induction, it follows that the value function is not an explicit function of the time k. Thus, we get

$$V_\tau(\xi_k) = \min_{u_k} Q_\tau(\xi_k, u_k), \tag{19}$$

where

$$\begin{aligned} Q_\tau(\xi_k, u_k) = &(\phi_k^{\mathrm{T}} \hat{\theta}_{k+1|k})^2 + \phi_k^{\mathrm{T}} P_{k+1|k} \phi_k + \sigma^2 \\ &+ \int_{-\infty}^{\infty} \varphi(X) V_{\tau-1}(\xi_{k+1}(\xi_k, u_k; X))\, \mathrm{d}X, \end{aligned} \tag{20}$$

and the optimal control at time k is given by

$$u_k^* = \operatorname*{argmin}_{u_k} Q_\tau(\xi_k, u_k). \tag{21}$$

The main problem with this recursive solution is that $V_\tau(\xi)$ for $\tau = 1, \ldots, T-1$ has to be computed according to (19) for all possible values of ξ, to be able to calculate the integral in the following recursion step. Since this cannot easily be carried out, a solution is to compute $V_\tau(\xi^i)$ for some finite set of points $\{\xi^i\}_{i=1}^N$ within a subset

$$\Omega = \left\{ \xi = (x, \theta, P) \mid \xi_{\min} \leq \xi \leq \xi_{\max}, \left|P_{ij}\right| = \left|P_{ji}\right| \leq \sqrt{P_{ii}P_{jj}} \right\} \tag{22}$$

of the hyperstate space and then use some suitable method to find an approximation $\widehat{V}_\tau(\xi)$ of $V_\tau(\xi)$ for arbitrary ξ by interpolating between these values. The limits in Ω should be chosen such that they include the parts of the space where the hyperstate is likely to be. The conditions on P follow since it is a covariance matrix.

In conclusion, we have constructed a method for determining the optimal control u_k^* with respect to the objective function (4) given the current hyperstate $\xi_k = (x_k, \hat{\theta}_{k+1|k}, P_{k+1|k})$, i.e., we can compute a policy function π_T such that

$$u_k^* = \pi_T(\xi_k). \tag{23}$$

Algorithm 1 summarizes the computation of the approximately optimal dual control law.

The hyperstate is obtained by starting from the initial state x_0 and prior parameter distribution $\hat{\theta}_{0|-1} = \hat{\theta}_0$ and $P_{0|-1} = P_0$, from which the initial hyperstate $\xi_0 = (x_0, \hat{\theta}_{1|0}, P_{1|0})$ can be computed with (8) and (9). It is then, in each step $k > 0$, updated by computing x_k from x_{k-1} and the last input and measurement, and again computing $\hat{\theta}_{k+1|k}$ and $P_{k+1|k}$ by (8) and (9).

Algorithm 1 Control Law Computation

Input: Time horizon T, system size n, m, noise parameters A, σ, R_v, and bounds $\xi_{\min}$, $\xi_{\max}$, $u_{\min}$, $u_{\max}$

Output: Value function approximation $\widehat{V}_T(\xi)$ and control law $u^* = \pi_T(\xi)$

1: **Initialize:** $\widehat{V}_0(\xi) = V_0(\xi) \equiv 0$
2: **for** τ = 1, 2, …, T **do**
3: Generate N hyperstate-space points $\{\xi^i\}_{i=1}^N$ from the set Ω in (22)
4: Set up expression for $\xi_{k+1}(\xi_k, u_k; X)$ according to (15), (16) and (17)
5: Compute $\{V_\tau(\xi^i)\}_{i=1}^N$ by minimizing $Q_\tau(\xi, u)$ according to (19) and (20) with $V_{\tau-1}(\xi)$ replaced by $\widehat{V}_{\tau-1}(\xi)$
6: Create an approximation $\widehat{V}_\tau(\xi)$ of $V_\tau(\xi)$ for arbitrary ξ by interpolating between the points $\{V_\tau(\xi^i)\}_{i=1}^N$
7: **end for**
8: By using $\widehat{V}_{T-1}$ in place of V_{T-1}, define the function $\pi_T(\xi) = \operatorname{argmin}_u Q_T(\xi, u)$ according to (20)
9: **return** $\widehat{V}_T(\xi)$, $\pi_T(\xi)$

3.3 Cautious and Certainty Equivalence Control

Due to the fact that $V_0 \equiv 0$, the optimal control for time horizon $T = 1$ can by (19) and (20) be computed analytically. To facilitate this, we introduce a partitioning of the parameter vector

$$\theta_k = \begin{bmatrix} b_k & \alpha_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{24}$$

where $b_k := b_k^1$ and

$$\alpha_k := \begin{bmatrix} a_k^1 \dots a_k^n & b_k^2 \dots b_k^m \end{bmatrix}^{\mathrm{T}}. \tag{25}$$

The covariance matrix is accordingly partitioned

$$P_k = \begin{bmatrix} P_k^b & P_k^{b\alpha} \\ (P_k^{b\alpha})^{\mathrm{T}} & P_k^{\alpha} \end{bmatrix}. \tag{26}$$

The Bellman equation (19) then gives the value function $V_1(\xi)$ as the minimum of

$$\begin{aligned} Q_1(\xi_k, u_k) &= u_k^2 \hat{b}_{k+1|k}^2 + x_k^{\mathrm{T}} \hat{\alpha}_{k+1|k} \hat{\alpha}_{k+1|k}^{\mathrm{T}} x_k \\ &+ 2u_k \hat{b}_{k+1|k} \hat{\alpha}_{k+1|k}^{\mathrm{T}} x_k + 2u_k P_{k+1|k}^{b\alpha} x_k \\ &+ u_k^2 P_{k+1|k}^b + x_k^{\mathrm{T}} P_{k+1|k}^{\alpha} x_k + \sigma^2, \end{aligned} \tag{27}$$

and the optimal control u_k^* as the minimizer. As long as $\hat{b}_{k+1|k} \neq 0$, we can minimize this by differentiating with respect to u_k and requiring the derivative to be equal to zero, giving the minimizing input

$$u_k^{\mathrm{cc}} = -\frac{\hat{b}_{k+1|k} \hat{\alpha}_{k+1|k}^{\mathrm{T}} + P_{k+1|k}^{b\alpha}}{\hat{b}_{k+1|k}^2 + P_{k+1|k}^b} x_k. \tag{28}$$

This is the so-called *cautious control* policy. If the parameter b_{k+1} is uncertain, the control policy (28) takes the risk of erroneous control effects into account by reducing the control action u. If, on the other hand, the variance of all parameters is assumed to be zero, we get the control law

$$u_k^{\mathrm{ce}} = -\frac{\hat{\alpha}_{k+1|k}^{\mathrm{T}}}{\hat{b}_{k+1|k}} x_k. \tag{29}$$

This is the *certainty equivalence* control, which is optimal if the parameters are exactly known.

The fact that the cautious controller reduces the input when the uncertainty is large might be advantageous, but might also cause problems. If the control action is too small, the uncertainty of the parameter values might increase instead of decrease, which leads to even smaller control action, until the control becomes zero. This is the so-called *turn-off phenomenon*, which can be resolved by probing introduced by e.g. solving the dual control problem for a longer time horizon.

4. Simulations

Here, we present results obtained when applying the algorithm with a specific method for evaluation point selection and interpolation to some simple systems. In each recursion step of Algorithm 1, N hyperstate-space points $\{\xi^i\}_{i=1}^N$ are chosen by uniform sampling from the subset Ω in (22). It is also assumed that the control signal is limited to $|u_k| \leq 100$ by an added saturation.

For the interpolation, a neural network is used. It consists of 5 dense inner layers with ReLU activation functions and 32 nodes each. Also the input layer has ReLU activation, while the one-node output layer does not have any activation function. As training loss, the mean squared error is used, and the ADAM optimizer is used with learning rate 10^{-4}. At least 100 training epochs are carried out, until the training loss is less than 0.01 or until the number of epochs is 1000.

For each system, 1000 different realizations are simulated, each with the length of 100 time steps. In the simulations, the computed control is applied as long as the hyperstate is inside the set Ω. If the hyperstate leaves this set, cautious control (28) is applied until the hyperstate returns to Ω. The result is compared with the costs obtained when controlling the system with alternative control policies while using the same noise sequence realizations $\{e_k\}$ and $\{v_k\}$. The simulation cost is defined as $\sum_{k=1}^T y_k^2$.

4.1 Example 1

The algorithm is applied to the system

$$y_{k+1} = y_k + bu_k + e_{k+1}, \tag{30}$$

studied in [Åström and Helmersson, 1986], where b is constant but unknown, and has an initial distribution $b \sim \mathcal{N}(\hat{b}_{1|0}, P_{1|0})$. For this simple case, a normalized hyperstate can be described by only two variables, which makes it easy to evaluate the value function on a regular grid using traditional methods. We use this to compare with the neural network approximation method proposed above. As shown in [Åström and Helmersson, 1986], it is sufficient to consider the normalized hyperstate $\xi_k = (\eta_k, \beta_k)$, where

$$\eta_k := y_k/\sigma \quad \text{and} \quad \beta_k := \hat{b}_{k+1|k}/\sqrt{P_{k+1|k}}. \tag{31}$$

The sampling set is chosen as

$$\Omega = \{\xi = (\eta, \beta) \mid -2 \leq \eta \leq 2,\ -4 \leq \beta \leq 4\} \tag{32}$$

and $N = 100$ points are sampled in each recursion step. Furthermore, we set $\sigma = 1$ and get $A = 1$ and $R_v = 0$ since the parameter is constant, i.e. $b_{k+1} = b_k$. The initial hyperstate is chosen as

$$x_0 = y_0 = 1, \quad \hat{b}_{1|0} = 0.7, \quad P_{1|0} = 1, \tag{33}$$

Table 1. Average performance cost $\sum_{k=1}^{100} y_k^2$ (with sample standard deviations in parenthesis).

Control Policy	Example 1	Example 2
Dual Control, NN	158 (29)	385 (384)
Dual Control, Linear	132 (20)	1694 (571)
Certainty Equivalence	153 (548)	29356 (34491)
Cautious Control	113 (36)	676 (351)
Optimal (θ known)	101 (14)	102 (14)

while the true parameter is $b = 1$. The time horizon for the loss function is chosen as $T = 30$, since it is shown in [Åström and Helmersson, 1986] that the change of the value function for a larger horizon is small. The control policy is therefore assumed to be close to optimal for a longer time horizon as well, such as the 100-step horizon that is simulated.

As reference for the result, the dual control is computed by evaluating the value function in each recursion step on an equidistant 100×100 grid on the $\eta\beta$-plane for $0.01 \leq \eta \leq 2$ and $0.01 \leq \beta \leq 4$. For this particular example, the values for negative variables are given by the symmetry of the value function and therefore only positive variable values have to be considered. This fact is, however, not used when applying the main algorithm.

The value function approximation from the main algorithm and the reference from the linear interpolation algorithm are shown in Fig. 1. The approximation has a maximal relative error of around 6%, even though the value function was evaluated in only 100 points per recursion step, compared with the 10 000 points for the reference solution.

The results from the simulations are shown in Tab. 1. The control policies are, in order, the one obtained from the suggested algorithm with neural network interpolation (Dual Control, NN), the one obtained as in [Åström and Helmersson, 1986], with the parameter assumed to be constant and using linear interpolation (Dual Control, Linear), certainty equivalence control (29), cautious control (28), as well as the optimal minimum variance control if the parameters would have been known (Optimal), which is (29) with the real parameters instead of the means.

For this particular example, with initialization (33) and real parameter $b = 1$, the cautious control gives a close-to-optimal cost. However, this does not apply in general, and different settings give a significantly higher cost with this method. The NN algorithm has, for the studied case, an average cost comparable to the CE control, but with significantly less variance, and seems therefore to be more robust to disturbances.

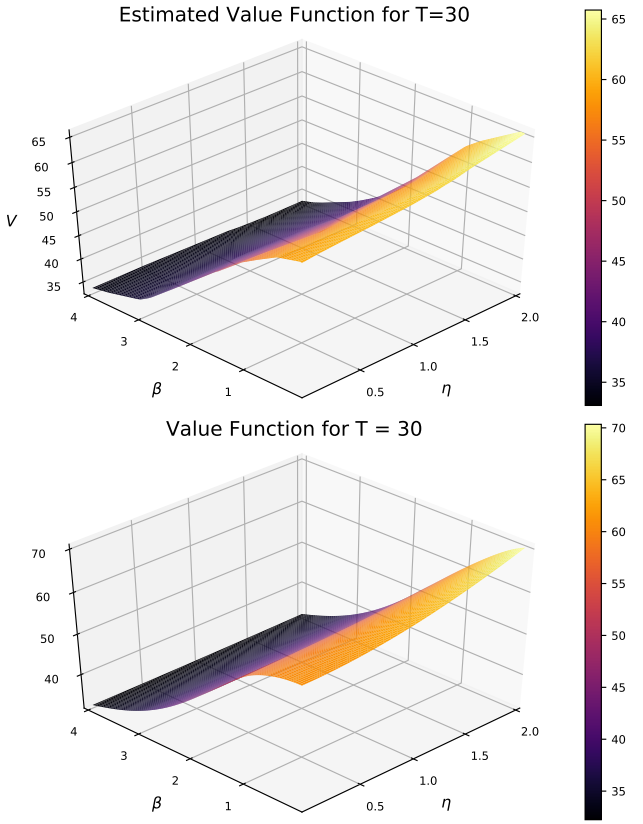


Figure 1. The estimated value function V_{30} in Example 1 and the same value function computed by linear interpolation. The neural network gives a good representation of the true value function.

4.2 Example 2

Now, a time-varying system with a similar structure is considered

$$y_{k+1} = y_k + b_{k+1}u_k + e_{k+1}, \tag{34}$$

where $b_1 \sim \mathcal{N}(\hat{b}_{1|0}, P_{1|0})$ and where

$$b_{k+1} = 0.9b_k + v_k \tag{35}$$

with $R_v = 0.19$, giving a unit-variance b. The sampling set for the 3-dimensional hyperstate is chosen as

$$\Omega = \{\xi = (y, \hat{b}, P) \mid |y| \leq 10, |\hat{b}| \leq 4, 0 \leq P \leq 2\} \tag{36}$$

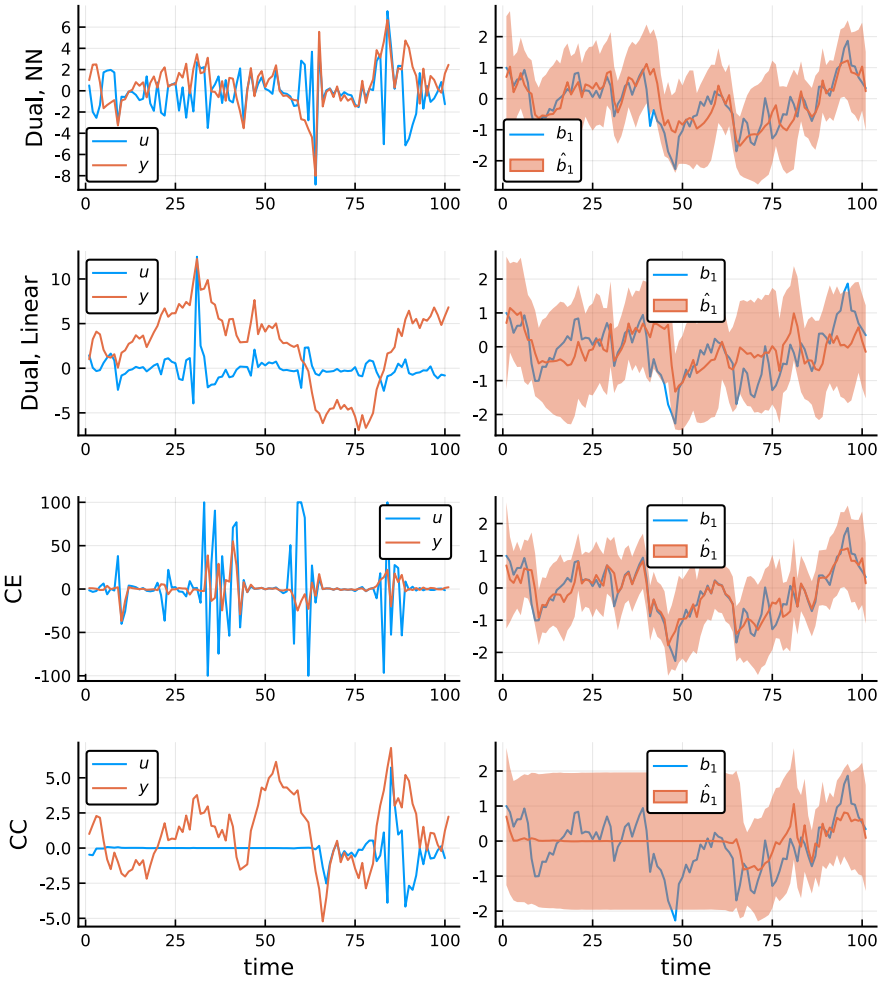


Figure 2. Resulting control, output and parameter estimation for one of the simulations from Example 2, using the control from the presented algorithm (Dual, NN), the dual controller computed for a system with constant parameters and linear interpolation (Dual, Linear), certainty equivalence control (CE) and cautious control (CC). A 95% confidence interval is shown for the parameter estimation.

and $N = 100$ points are sampled in each recursion step. Initial values are again chosen as (33), and the true initial parameter value is $b_1 = 1$.

The resulting costs, with sample standard deviations, are shown in Tab. 1. In this case, the suggested controller with the particular initialization gives better performance than all the other controllers. For a slow parameter variation, the Linear method could possibly perform well, but for this example, the constant-parameter assumption leads to a rather high cost. One of the simulations is shown in Fig. 2. The Linear method gives a worse parameter estimation than the NN method, in terms of mean and variance. The CE gives huge controls (which are saturated at ± 100) when the parameter mean is close to zero, which causes large peaks in the output. Finally, we note the turn-off phenomenon for the cautious controller CC, where the control is set to zero and the variance remains large during a time interval.

5. Conclusions

We have described an algorithm for approximate computation of the optimal dual control law for a discrete-time linear system with time-varying parameters. The algorithm can be used with any appropriate method for selecting evaluation points for the value function and for interpolating between these points. By studying an example, we have shown that this can lead to improved control performance relative to a dual control method assuming constant parameters, as well as two standard methods from adaptive control. In the example, we have used the simplest possible neural network type for interpolation; a network with only dense layers. It has been trained by only 100 values in each recursion. For larger examples, a more advanced structure would probably be needed, but we see it as promising that even this simple structure can perform relatively well. With the wide variety of different structures and parameters that could be used for a neural network, there should be excellent opportunities to conduct further investigations in order to improve the approach and apply it to more advanced settings. Alternative ways of interpolation as well as evaluation point selection are also promising subjects for future research.

6. Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Åström, K. J. and A. Helmersson (1986). "Dual control of an integrator with unknown gain". *Comput. Math. with Appl.* **12**:6, Part A, pp. 653–662. ISSN: 0898-1221. DOI: https://doi.org/10.1016/0898-1221(86)90052-0.

Åström, K. J. and B. Wittenmark (2011). *Computer-controlled systems: theory and design*. 3rd ed. Dover Publications Inc., p. 431.

Bayard, D. and A. Schumitzky (2010). "Implicit dual control based on particle filtering and forward dynamic programming". *Int J Adapt Control Signal Process* **24**:3, pp. 155–177.

Feldbaum, A. A. (1960). "Dual control theory I-II". *Autom. Remote Control* **21**, pp. 874–880, 1033–1039.

Feldbaum, A. A. (1961). "Dual control theory III-IV". *Autom. Remote Control* **22**, pp. 1–12, 109–121.

Klenske, E. D. and P. Hennig (2016). "Dual control for approximate Bayesian reinforcement learning". *Journal of Machine Learning Research* **17**, pp. 1–30. ISSN: 15337928. arXiv: 1510.03591.

Lindoff, B., J. Holst, and B. Wittenmark (1998). "Analysis of Suboptimal Dual Control". *IFAC Proceedings Volumes* **31**:22, pp. 21–28. ISSN: 14746670. DOI: 10.1016/s1474-6670(17)35915-3.

Recht, B. (2019). "A Tour of Reinforcement Learning: The View from Continuous Control". *Annu. Rev. Control Robot. Auton. Syst.* **2**:1, pp. 253–279. ISSN: 2573-5144. DOI: 10.1146/annurev-control-053018-023825. arXiv: 1806.09460.

Sutton, R. and A. Barto (2018). *Reinforcement Learning: An Introduction (2nd ed., in preparation)*. MIT Press.

Svensson, A. (2016). *Learning probabilistic models of dynamical phenomena using particle filters*. Licentiate Thesis. Uppsala University.

Tse, E. and Y. Bar-Shalom (1973). "An actively adaptive control for linear systems with random parameters via the dual control approach". *IEEE Transactions on Automatic Control* **18**:2, pp. 109–117.

Tse, E., Y. Bar-Shalom, and L. Meier (1973). "Wide-sense adaptive dual control for nonlinear stochastic systems". *IEEE Transactions on Automatic Control* **18**:2, pp. 98–108.

Wittenmark, B. (1975). "An Active Suboptimal Dual Controller for Systems with Stochastic Parameters". *Automatic Control Theory and Application* **3**, pp. 13–19.

Wittenmark, B. (1995). "Adaptive Dual Control Methods: An Overview". *IFAC Proceedings Volumes* **28**:13, pp. 67–72. ISSN: 14746670. DOI: 10.1016/s1474-6670(17)45327-4.

Paper II

Dual Control by Reinforcement Learning Using Deep Hyperstate Transition Models

Christian Rosdahl Anton Cervin Bo Bernhardsson

Abstract

In dual control, the manipulated variables are used to both regulate the system and identify unknown parameters. The joint probability distribution of the system state and the parameters is known as the hyperstate. The paper proposes a method to perform dual control using a deep reinforcement learning algorithm in combination with a neural network model trained to represent hyperstate transitions. The hyperstate is compactly represented as the parameters of a mixture model that is fitted to Monte Carlo samples of the hyperstate. The representation is used to train a hyperstate transition model, which is used by a standard reinforcement learning algorithm to find a dual control policy. The method is evaluated on a simple nonlinear system, which illustrates a situation where probing is needed, but it can also scale to high-dimensional systems. The method is demonstrated to be able to learn a probing technique that reduces the uncertainty of the hyperstate, resulting in improved control performance.

Originally published in the 14th IFAC Workshop on Adaptive and Learning Control Systems ALCOS 2022, Casablanca, Morocco, June 29 – July 1, 2022. Reprinted with permission.

1. Introduction

When controlling dynamical systems whose characteristics are not fully known, the controller needs to balance between achieving good control quality based on current information and experimenting with the system to facilitate better future control. Research into such *dual controllers* goes back to the work of [A. A. Feldbaum, 1960], who described that control inputs to uncertain systems should have a *probing effect* for active learning and uncertainty reduction, in addition to actively regulating the system dynamics. For surveys of dual control theory, see [Wittenmark, 1995] and [Mesbah, 2016]. The same problem formulation occurs in the area of Bayesian reinforcement learning, see [Klenske and Hennig, 2016] and the references therein. For a survey linking the field with recent advances in reinforcement learning, see [Matni et al., 2019].

Various methods have been proposed to achieve this so-called exploration–exploitation trade-off. From a theoretical perspective, the situation can, as Feldbaum showed, be formulated as a traditional optimal control problem and solved through dynamic programming using a Bellman functional equation, see [Åström, 1965; Bertsekas, 1987]. However, this requires a representation of the system using the conditional probability of the extended system state – the so-called *hyperstate* – consisting of the probability distribution of the state and the parameters of the system. This is usually computationally intractable. Explicitly incorporating model uncertainty into an optimal stochastic control problem, and minimizing a one-step-ahead error, leads to *myopic* control, also called *cautious control*, because control inputs will typically become small (cautious) when the uncertainty is large [Wittenmark, 1995]. Small inputs will, in turn, generate less information about the system, which might eventually lead to a so-called turn-off phenomenon.

There are many suggestions on how to encourage exploration, a popular choice being a simple ε-greedy approach, where the control signal is chosen randomly with probability ε at each time instance. A more structured approach is to extend the goal function with a penalty reflecting model uncertainty, see, e.g. [Filatov and Unbehauen, 2000]. An approximate dual control for linear systems is constructed in [Tse et al., 1973; Tse and Bar-Shalom, 1973] using a quadratic expansion around a nominal trajectory to describe the cost of small parameter uncertainties, assuming that the hyperstate can be described by a Gaussian distribution. The nominal trajectory is constructed from the mean system using the certainty equivalence principle. The method is extended to a nonparametric Gaussian process dynamics model in [Klenske and Hennig, 2016], allowing for nonlinear features in the system dynamics. Examples are provided for the use of the framework describing the uncertain system dynamics as either an (approximate) Gaussian process or a feedforward neural network, producing structured exploration strategies. Furthermore, in recent years, several suggestions have been presented on how to incorporate dual control elements into the model-predictive control (MPC) framework, for example, in [Heirung et al., 2017; Soloperto et al., 2019; Parsi et al., 2020; Arcari et al., 2020].

The computation of optimal dual control policies requires solving stochastic dynamic programming equations with a hyperstate of high (often infinite) dimension $\mathbb{P}(x_k, \theta_k \mid \mathcal{D}_k)$ where $x, \theta, \mathcal{D}$ denote the state, parameters, and available data, respectively. Solutions have been computed only for small toy examples, and scaling up to realistic problems requires overcoming the "curse of dimensionality" in representing the evolution of the hyperstate,

$$\mathbb{P}(x_{k+1}, \theta_{k+1} \mid \mathcal{D}_{k+1}) = \mathcal{T}\left(\mathbb{P}(x_k, \theta_k \mid \mathcal{D}_k), y_{k+1}, u_k\right), \tag{1}$$

where y_{k+1} and u_k are the most recent measurement and control signals, respectively. The operator $\mathcal{T}$ is unfortunately often intractable since it involves multidimensional integrals. Nonlinear estimation methods have been developed to find useful approximations, such as extended Kalman filters, Gaussian sum filters, and the particle filter; see, e.g., [Sorenson and Alspach, 1971; Arulampalam et al., 2002]. The paper [Bayard and Schumitzky, 2010] also describes a sampling-based approach to dual control that combines a particle filter with a policy iteration method for forward dynamic programming. Simulation results indicate that such methods can systematically improve closed-loop performance compared to other more common stochastic control approaches.

In this paper, we propose a method for representing the hyperstate as a mixture model and approximating the nonlinear hyperstate transition mapping using a deep neural network. The method assumes that we have a model of the system, with uncertain state and/or parameters that can be used for simulations to generate training data, as well as probability distributions of the system parameters and the initial state. Once the hyperstate transition model has been trained with the simulated data, it is used in combination with a reinforcement learning algorithm to train a policy, which then can be used for carrying out dual control on the real system.

The rest of the paper is outlined as follows. In Sec. 2, the dual control problem is formulated. Sec. 3 describes our hyperstate transition model. In Sec. 4, we describe an example similar to that proposed in [Alspach, 1972], illustrating that interesting issues of dual control can arise even for a simple scalar system with known dynamics, as long as the measurement equation is nonlinear. Sec. 5 presents results for the hyperstate transition model training in the example. Sec. 6 describes the reinforcement learning algorithm used, and Sec. 7 shows results from using the hyperstate transition model for control, both directly and in combination with reinforcement learning to find a dual control law. Sec. 8 concludes the paper.

2. The Dual Control Problem

A general discrete-time system with state x_k, control signal u_k, and measurement signal y_k is given in the form

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, w_k; \theta_k^f), \\ y_k &= g(x_k, e_k; \theta_k^g), \end{aligned} \tag{2}$$

where $\theta_k = (\theta_k^f, \theta_k^g)$ are time-varying parameters and w_k and e_k are disturbances with known stochastic properties. We consider the problem of finding an optimal control policy π that minimizes some cost J. The policy can use past inputs and current and past outputs given by

$$\mathcal{D}_k = \{y_k, u_{k-1}, y_{k-1}, u_{k-2}, \dots\}. \tag{3}$$

The information that is relevant to determine the progression of the system is the current probability distribution of the state and the parameters of the system, conditioned on the available data $\mathcal{D}_k$. This probability measure is named *hyperstate* and is denoted by

$$\xi_k(A) = \mathbb{P}((x_k, \theta_k) \in A \mid \mathcal{D}_k). \tag{4}$$

It is known that an optimal control policy depends on the history $\mathcal{D}_k$ solely through the hyperstate ξ_k [Bertsekas, 1987]. Therefore, the dual control problem can be formulated as finding a policy $\pi(\xi_k)$ that minimizes the cost

$$J_k(\pi, \xi_k) = \mathbb{E}\left\{ \sum_{i=k}^{k+T-1} c_i(x_{i+1}, u_i) \,\middle|\, \xi_k \right\}, \tag{5}$$

where $u_i = \pi(\xi_i)$ is the control action, $c_i(x_{i+1}, u_i)$ is the step cost, and T is some time horizon. The hyperstate ξ_0 represents initial knowledge of the state and the parameters.

In principle, the optimal control policy can be computed by dynamic programming, but due to the non-Gaussianity of the hyperstate and its nonlinear evolution, the problem is in general intractable even for low-dimensional, fixed-parameter systems with Gaussian disturbances, since the hyperstate, describing a continuous probability measure, in general is an infinite-dimensional object. Hence, approximate methods are required for anything but toy examples, with the LQG control problem being a notable exception.

3. Hyperstate Transition Model

In general, the hyperstate is a highly complex and multivariate probability distribution. Therefore, it is often intractable to calculate it and update it at each time step, when $\mathcal{D}_k$ is replaced by $\mathcal{D}_{k+1}$. For this reason, we want to find a way to compactly represent an approximation of the probability distribution using a relatively small set of parameters. We can then devise a method to quickly and efficiently update this compact representation at each time step, using the newly obtained data.

3.1 Hyperstate Representation

We will assume that the hyperstate ξ_k can be approximately described by a mixture model consisting of a convex combination of c components f_i, each with a parame-

ter vector β_k^i and weighted by λ_k^i, i.e., a probability density

$$f_{\xi_k}(\xi) = \sum_{i=1}^{c} \lambda_k^i f_i(\xi; \beta_k^i). \tag{6}$$

The hyperstate can then be represented by the vector

$$\alpha_k = \begin{bmatrix} \lambda_k^1 & \dots & \lambda_k^{c-1} & (\beta_k^1)^{\mathrm{T}} & \dots & (\beta_k^c)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{7}$$

In this paper, we use Gaussian mixture models, where parameters β_k^i capture the mean and covariances of the components f_i at time k. Given samples $\{\xi_k^i\}_{i=1}^n$ from the hyperstate, the vector α_k can easily be estimated by the *Expectation Maximization (EM) algorithm*, see Ch. 8.5 in [Hastie et al., 2001]. The number of components c is treated as a user parameter and is here, for simplicity, held fixed.

Our goal is to construct a computationally efficient *hyperstate transition model (HTM)*

$$\alpha_{k+1} = \mathcal{T}_{\mathrm{HTM}}(\alpha_k, u_k, y_{k+1}), \tag{8}$$

which updates the hyperstate representation when new data (u_k, y_{k+1}) become available. In this paper, we will use neural networks that have a good ability to represent complex nonlinear mappings and, once trained, are able to compute the mapping efficiently. This is essential to later be able to train a dual control policy in an efficient way.

3.2 Simulation to Obtain HTM Training Data

To train the HTM neural network, we use the system model to simulate a number of episodes using a suitable controller (more on the choice of this later). At time step k in an episode, we use α_k, assumed known, to sample a set of values $\{\xi_k^i\}$ from the corresponding mixture model. Each of these values is propagated through the system model, together with the same input u_k as in the main simulation. The resulting outputs $\{\xi_{k+1}^{\mathrm{pri},i}\}$ then correspond to samples of the prior distribution at time $k+1$. By sampling (with replacement) from this set, with weights corresponding to the probability of obtaining the measurement y_{k+1} given the state and parameters prescribed by each ξ_k^i, we obtain a set of values $\{\xi_{k+1}^i\}$ corresponding to the hyperstate samples at time $k+1$. Fitting the parameters of the mixture model to these values gives us the new target vector α_{k+1}. The initial α_0 is obtained by sampling $\{\xi_0^{\mathrm{pri},i}\}$ from some initial assumption of the prior hyperstate, and applying weighted sampling by the initial measurement y_0 in a similar fashion. The procedure is illustrated in the top part of Fig. 1.

3.3 Model training and application

The data from the simulation are used to train the HTM neural network to produce an estimate $\hat{\alpha}_{k+1}$ of α_{k+1} given $(\hat{\alpha}_k, u_k, y_{k+1})$, as shown in the middle part of Fig. 1.

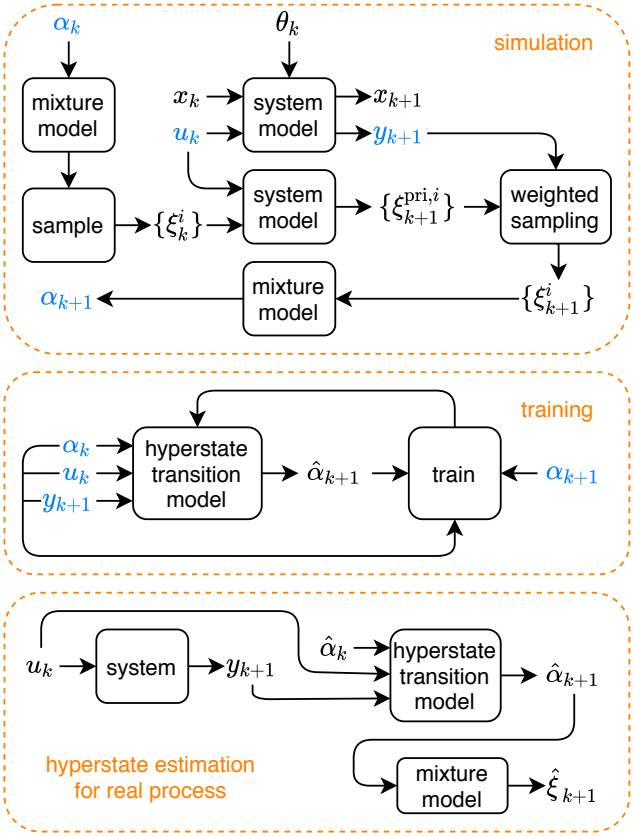


Figure 1. Illustration of the HTM construction procedure (top two boxes) and application of the HTM (bottom box). For description, see Algorithm 1.

The method is summarized in Algorithm 1. The HTM can now be used with the real system, or with another simulation of it, to quickly give an estimate $\hat{\xi}_{k+1}$ of the current hyperstate, yielding an estimate $\hat{\alpha}_{k+1}$ of the parameter vector for the mixture model, see the lower part of Fig. 1.

4. Example System

The suggested method will be illustrated on an integrator system with a nonlinear measurement function, given by

$$\begin{aligned} x_{k+1} &= x_k + u_k + w_k, \\ y_k &= |x_k| + e_k, \end{aligned} \tag{9}$$

Algorithm 2 Hyperstate Transition Model – Construction

Input: Number of simulation episodes E and time steps T per episode, number of hyperstate sample points n, initial hyperstate estimate $\hat{\xi}_0^{\text{pri}}$, and neural network structure and training settings

Output: Model for estimating α_{k+1} from (α_k, u_k, y_{k+1})

for all episodes **do**
  for $i \leftarrow 1$ to n **do**
    Sample $\xi_0^{\text{pri},i}$ from $\hat{\xi}_0^{\text{pri}}$
    Compute weight $w_i = \mathbb{P}(y_0 \mid \xi_0^{\text{pri},i})$
  end for
  Sample $\{\xi_0^i\}_{i=1}^n$ from $\{\xi_0^{\text{pri},i}\}$ with weights $\{w_i\}$
  Get α_0 by fitting mixture model to $\{\xi_0^i\}$
  for $k \leftarrow 0$ to $T-1$ **do**
    Select u_k and get y_{k+1} from simulation
    Sample $\{\xi_k^i\}_{i=1}^n$ from mixture model with α_k
    for $i \leftarrow 1$ to n **do**
      Get $\xi_{k+1}^{\text{pri},i}$ from simulation with u_k
      Compute weight $w_i = \mathbb{P}(y_{k+1} \mid \xi_{k+1}^{\text{pri},i})$
    end for
    Sample $\{\xi_{k+1}^i\}_{i=1}^n$ from $\{\xi_{k+1}^{\text{pri},i}\}$ with weights $\{w_i\}$
    Get α_{k+1} by fitting mixture model to $\{\xi_{k+1}^i\}$
  end for
end for
Train neural network with inputs $\{(\alpha_k^e, u_k^e, y_{k+1}^e)\}$ and outputs $\{\alpha_{k+1}^e\}$ from each episode e and time step k

where the process disturbance w_k and the measurement noise e_k are independent white noise processes with given probability density functions. A similar setup was considered in [Alspach, 1972], with a quadratic measurement equation.

Note that this system does not contain any unknown parameters θ_k, so the hyperstate is given by the probability distribution $\xi_k = \mathbb{P}\{x_k \mid \mathcal{D}_k\}$. Probing will still be an essential element of any useful control policy; to find the sign of x_k, one needs to use a nonzero control u_k, leading to a dual control behavior.

For evaluation of the methodology, we considered two different signal models. First, we assume a discrete-valued model, where w_k and e_k have identical discrete probability functions $p_{w,e}(n) := \mathbb{P}(w_k = n) = \mathbb{P}(e_k = n)$ with $p_{w,e}(0) = 0.3$, $p_{w,e}(\pm 1) = 0.2$, $p_{w,e}(\pm 2) = 0.1$, $p_{w,e}(\pm 3) = 0.05$, and $p_{w,e}(n) = 0$ for all other n. Furthermore, u_k and x_k can only assume integer values between -10 and 10. Therefore, the hyperstate can be represented by a 21-dimensional vector, which simplifies the verification of the implementation.

Second, we assume a continuous-valued model with $w_k \sim \mathcal{N}(0, 0.1)$, $e_k \sim$

$\mathcal{N}(0,1)$ ($\mathcal{N}$ denoting a normal distribution) and $u_k \in \mathbb{R}$. Here, the hyperstate cannot be represented by a finite-dimensional vector, but since the system is scalar, an arbitrarily fine approximation of the hyperstate probability distribution can be computed for verification.

5. Hyperstate Transition Model – Results

We trained the HTM network for (9) for both the discrete-valued and continuous-valued signal models, both of which give good results. Here, we show only the results for the discrete-valued model, while both models were used in the control experiments in Sec. 7.

For the HTM network, we experimented with different architectures, and finally settled on a feedforward network with six inner dense layers, each with 64 nodes and ReLU activation functions, and a dense layer without activation function as output layer. After each inner layer, except for the first, we used a dropout layer with a dropout ratio 0.3 for regularization. The inputs and outputs of the model were normalized to zero mean and unit variance.

To generate training and test data, the example system (9) was simulated for $E = 10^3$ episodes, each with $T = 10$ time steps. For each simulation step, we sample $n = 10^3$ points from the hyperstate to estimate the posterior distribution. The control signal is chosen as $u_k = -\hat{x}_k$, where $\hat{x}_k$ with 90% probability is $\hat{x}_k \sim f_{\xi_k}$, i.e., sampled from the current hyperstate estimate given by the mixture model, and with 10% chance is selected uniformly randomly from all possible state values. The idea with this control policy is that most of the time the system will behave closely to how it will behave when reasonably well controlled (with the goal of reaching $x = 0$), but that the simulation data will also contain examples of how the hyperstate changes for arbitrary inputs.

We used 800 of the simulated episodes as training data for the HTM network and 200 episodes as test data. The model was implemented in Python with Tensorflow and trained using 3000 epochs, a mean squared error loss, and the ADAM optimizer with a learning rate 10^{-3}.

We used $c = 2$ Gaussian components with parameters $\beta_k^i = (\mu_k^i, \sigma_k^i)^{\mathrm{T}}$ that describe means and standard deviations for the mixture model. Typical hyperstate representations for the first eight time steps of an episode are shown in Fig. 2, with the exact hyperstate as a reference. For this problem, increasing c beyond 2 gave no improvement.

To evaluate the HTM performance on the test data, we use the Wasserstein L_1 distance between the predicted and actual mixture model distributions. The Wasserstein distance describes the amount of "work" required to transform one probability distribution into another, in terms of the probability density that must be moved, multiplied by the distance to be moved, see [Villani, 2009]. Costs for the 2000 time steps in the test episodes were distributed as shown in Fig. 3. The average cost was

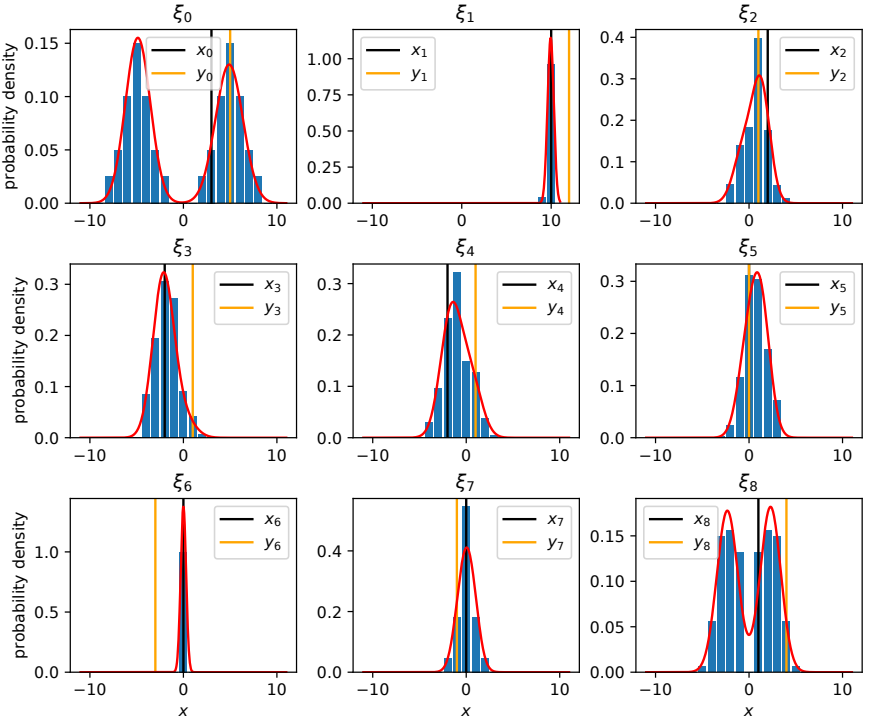


Figure 2. Hyperstate representations by the mixture model (red) for some time steps of one episode. Exact hyperstate (blue) is shown for reference. Actual state x_k (black) and measurement y_k (orange) are also shown.

0.41, which indicates a reasonably good fit. Some of the predictions for the test data, with varying Wasserstein costs (stated above each subplot), are shown in Fig. 4.

6. Reinforcement Learning Algorithm

Our goal is now to find a control policy $\pi(\xi_i)$ that minimizes a cost of the form (5). With T being the total time for which the system is run, we will use the cost

$$J_x(\pi, \xi_k) = \mathbb{E}\left\{ \sum_{i=k}^{T-1} x_{i+1}^2 \,\middle|\, \xi_k \right\}. \tag{10}$$

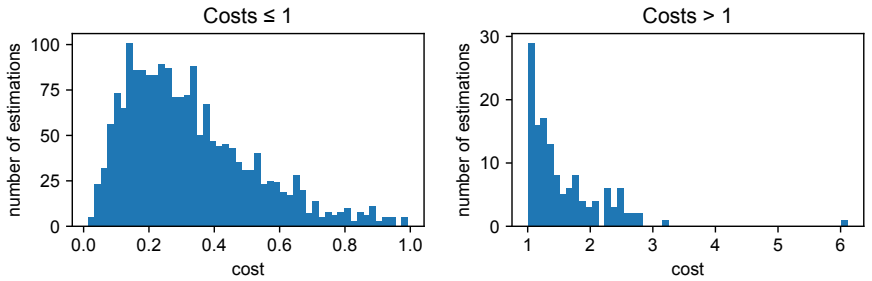


Figure 3. Histogram of Wasserstein distances between the HTM-predicted and actual mixture model distributions for all time steps in the test data.

Assuming that e_k is independent of e_l for $l \neq k$ and has zero mean, minimizing J_x is equivalent to minimizing

$$J_y(\pi, \xi_k) = \mathbb{E}\left\{ \sum_{i=k}^{T-1} y_{i+1}^2 \,\middle|\, \xi_k \right\}, \tag{11}$$

since $y_k = |x_k| + e_k$, and e_k is independent of ξ_k and x_k. Described in reinforcement learning (RL) terms, our state S_k, action A_k, and reward R_k at time k are

$$S_k = \alpha_k, \quad A_k = u_k, \quad R_k = -y_k^2, \tag{12}$$

where α_k represents the hyperstate ξ_k. The RL training aims to find a control signal u_k that maximizes the expected value of the return G_k, here chosen as

$$G_k = R_{k+1} + R_{k+2} + \cdots + R_{k+h}, \tag{13}$$

where the horizon h could be chosen different from T. To maximize the return, it might be optimal to select an action u_k that gives a smaller reward R_{k+1} in the next time step, but leads to the collection of more information so that the following $h-1$ rewards can be higher. We found that a horizon $h = 5$ was sufficient to achieve such an exploration effect for our simple example system (the cost of choosing a larger h increases linearly).

By RL we want to find an action–value function $Q(\alpha_k, u_k)$ that gives an estimate of the expected return G_k at time k, if the system is in state α_k, we select action u_k, and the following actions are selected according to an optimal policy π. A neural network is used to represent this function. The network weights ω_l are updated according to Algorithm 2. For a more detailed description of algorithms of this type and other alternatives, see [Sutton and Barto, 2018].

We assume that the system is run for E episodes, with T time steps per episode, with an initial hyperstate estimate at each episode described by some vector α_0.

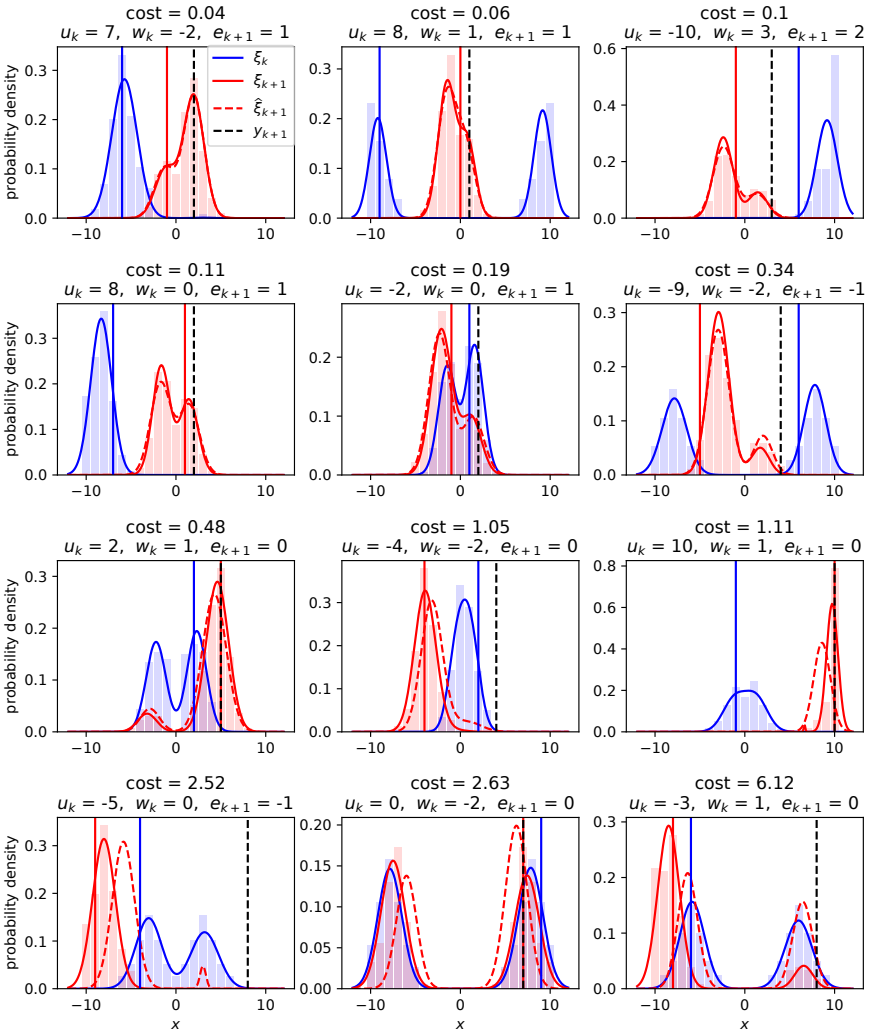


Figure 4. Some HTM-predictions for test data, with Wasserstein distance costs. Solid curves show correct hyperstate representations by the mixture model for current time step k (blue) and next time step $k+1$ (red). Dashed curves show hyperstate at time $k+1$ predicted by the HTM. Bars show the exact hyperstate at time k (blue) and time $k+1$ (red). Vertical lines show x_k (blue), x_{k+1} (red), and y_{k+1} (black dashed).

We also create a replay buffer $\mathcal{B}$ where we store the latest B states, actions, and rewards. When simulating, each action u_k is chosen according to an ε-greedy policy $\pi_t(\alpha_k)$, with α_k obtained from the HTM. The policy selects a random action u_k in the interval $[-10, 10]$ with probability ε_t and the action u_k that maximizes $Q(\alpha_k, u_k; \omega_t)$ with probability $1-\varepsilon_t$, where we let exploration decrease over time as $\varepsilon_t = \varepsilon_{\min} + (\varepsilon_{\max} - \varepsilon_{\min}) e^{-t/T_\varepsilon}$. At the end of each episode, the b tuples (α_k, u_k, G_k) are sampled from the replay buffer and used to update the weights ω_t of the Q network by moving in the direction of its gradient, with some step size η.

Algorithm 3 Reinforcement Learning Algorithm

Input: Number of episodes E and time steps T per episode, return horizon h, epsilon function ε_t, initial hyperstate representation α_0, replay buffer size B, batch size b, learning rate η, and neural network settings

Output: Action–value function $Q(\alpha_k, u_k)$

Build network $Q(\alpha_k, u_k; \omega_0)$ with random weights ω_0
Define an ε-greedy policy $\pi_t(\alpha_k) = \text{epsgreedy}(\varepsilon_t, Q)$
Initialize replay buffer $\mathcal{B}$ and total time count $t \leftarrow 0$
for all episodes **do**
    for $k \leftarrow 0$ to $T-1$ **do**
        Apply input $u_k = \pi_t(\alpha_k)$ and save R_{k+1}
        Get $\alpha_{k+1} = \mathcal{T}_{\text{HTM}}(\alpha_k, u_k, y_{k+1})$
        Update total time count $t \leftarrow t+1$
    end for
    for $k \leftarrow 0$ to $T-h$ **do**
        Compute return $G_k = \sum_{i=1}^{h} R_{k+i}$
        Add (α_k, u_k, G_k) to $\mathcal{B}$
    end for
    Sample tuples $\{(\alpha_k^i, u_k^i, G_k^i)\}_{i=1}^{b}$ from $\mathcal{B}$
    Update weights $\omega_{t+1} \leftarrow \omega_t + \eta \sum_{i=1}^{b} \left[G_k^i - Q(\alpha_k^i, u_k^i; \omega_t)\right] \nabla Q(\alpha_k^i, u_k^i; \omega_t)$
end for

In our tests, we used a network with 3 dense inner layers, each with 8 nodes and ReLU activation. The output layer is a dense layer of one node without any activation. The input consists of the values α_k and u_k, normalized by the scaling factors of the training data for the HTM network.

7. Control Results

7.1 Discrete-Valued System

We used the HTM to directly control the system and compare the result with some alternative control policies. Each policy was applied in $T = 10^4$ time steps. This was

Table 1. Mean and standard deviation for the cost – Discrete-valued system in Section 7.1

Control policy	**Cost** $J = \frac{1}{T}\sum_{i=1}^{T} y^2$
$u = 0$	40.7 ± 1.6
$u = \pm y$	25.5 ± 0.36
$u^{\mathrm{ML}} = -\max(\hat{\xi})$	9.82 ± 0.23
$u^{\mathrm{oracle}} = -x$	4.18 ± 0.05

repeated 10 times with results as reported in Table 1. The policy $u^{\mathrm{ML}} = -\max(\hat{\xi})$ corresponds to the optimal control for the most probable (maximum likelihood) state for the current hyperstate. We compare this with the zero control policies and $u = \pm y$, where we let u equal the current measurement signal but with a random sign (with equal probabilities). We also compare it with the optimal control $u^{\mathrm{oracle}} = -x$, if x had been known. For the tested policies, the one based on the hyperstate approximation gives the best cost out of the ones which do not require unavailable information.

7.2 Continuous-Valued System

For the continuous-valued model, we trained an HTM using the same settings as for the previous case. An action–value function Q was generated by Algorithm 2 and used for control. The result was compared to simple maximum-likelihood control, as described in the previous example.

We study a case in which each episode starts at $x_0 = 1$ and is of length $T = 10$. In this case, it is impossible for a controller to know whether a positive or negative control signal should be used to reduce the state error, and small controls do not lead to large enough changes to resolve the issue due to the measurement noise. Therefore, some exploration, using larger control signals, will be useful. We run the RL algorithm for $E = 4 \cdot 10^4$ episodes, with $\varepsilon_{\max} = 1$, $\varepsilon_{\min} = 10^{-2}$, $T_\varepsilon = 10^3$. For the last 5000 episodes, we set $\varepsilon_t = 0$, to obtain a Q function that approximates the action value for the optimal policy at the end of the simulation. The values in the initial hyperstate α_0 are chosen as $\lambda_0^1 = 0.5$, $\mu_0^1 = -1$, $\mu_0^2 = 1$, and $\sigma_0^1 = \sigma_0^2 = 1$, see Fig. 5. The replay buffer size is $B = 10^3$, and the batch size is $b = 10$. The weight update is carried out by the ADAM optimizer with a learning rate 10^{-3}. Since the input is one-dimensional and in a bounded range, the maximization of the Q function can be carried out by evaluating the function for all points on a grid over this range and then for some points on a denser grid around the maximizing point.

When the algorithm was run, the control cost $J_y = \frac{1}{T}\sum_{i=1}^{T} y^2$, including random exploration at the beginning of the simulation, was 3.51. We then used the final Q function to control the system for $E = 10^3$ episodes by selecting the maximizing u values as control inputs. When repeated 5 times, the mean cost was 1.97 ± 0.05.

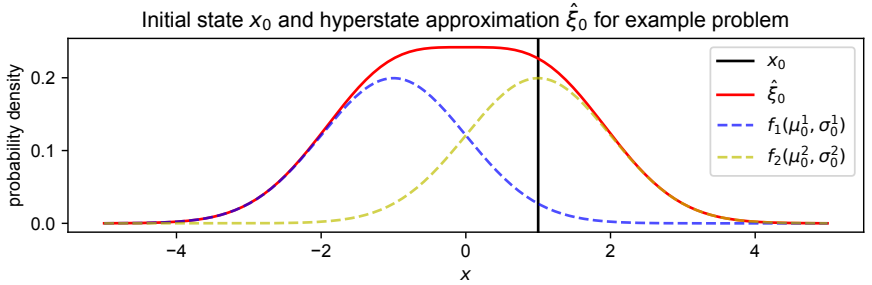


Figure 5. Initial state x_0 and hyperstate approximation $\hat{\xi}_0$ of each episode of the example problem, and the two mixture model components f_1 and f_2.

Table 2. Mean and standard deviation for the cost – Continuous-valued system in Sec 7.2

Control policy	**Cost** $J = \frac{1}{T}\sum_{i=1}^{T} y^2$
$u = 0$	2.45 ± 0.04
$u^{\mathrm{ML}} = -\max(\hat{\xi})$	2.46 ± 0.04
$u^{\mathrm{RL}} = \arg\max_u Q(\hat{\xi}, u)$	1.97 ± 0.05
$u^{\mathrm{oracle}} = -x$	1.20 ± 0.02

As a reference, we repeated the same test with the policy $u_k^{\mathrm{ML}} = -\max(\hat{\xi}_k)$, which gave relatively good results for the previous system. This gave the mean cost 2.46 ± 0.04. Thus, the suggested method manages to learn a policy that improves the cost through suitable exploration.

The signal values for 50 time steps, when the two different policies are run, are shown in Fig. 6. We have plotted $-u$ instead of u, since the optimal control $u = -x$ then corresponds to the overlapping lines. Note that the ML policy applies controls close to zero more frequently, failing to reduce state uncertainty. This is further illustrated in Fig. 7. The Q-function policy hence leads to better control performance by more efficient exploration.

8. Conclusion

The paper has proposed a novel design methodology for dual controllers aimed at optimizing the exploration–exploitation trade-off for uncertain systems. The first design step involves developing a neural network-based hyperstate transition model, which can be trained based on simulation data from a nonlinear process model. In

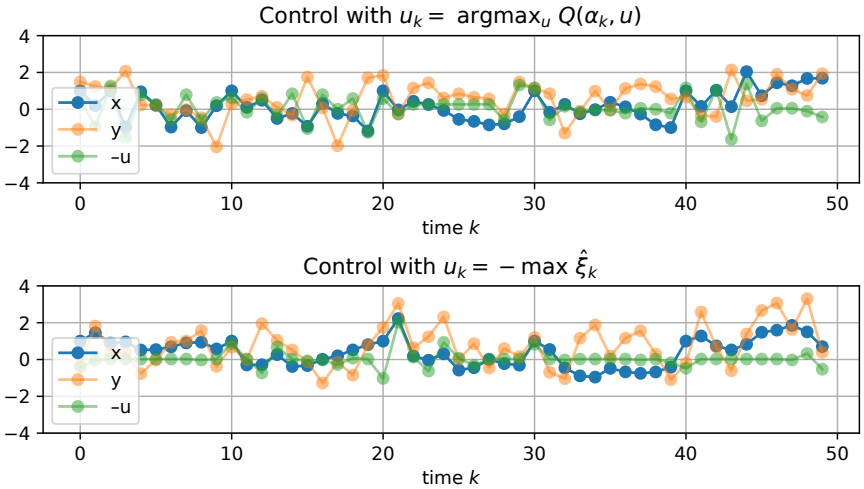


Figure 6. Typical behavior using $u_k = \arg\max_u Q(\alpha_k, u)$ with learned action–value function Q (top), or reference policy $u_k = -\max \hat{\xi}_k$ (bottom).

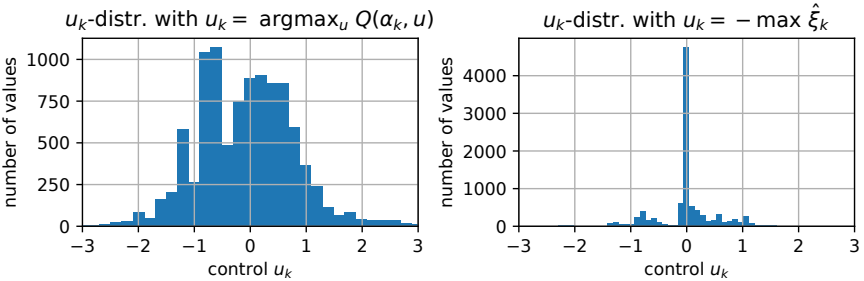


Figure 7. Control distributions for $u_k = \arg\max_u Q(\alpha_k, u)$ with the learned action–value function Q (left), or reference policy $u_k = -\max \hat{\xi}_k$ (right).

a second independent design step, the hyperstate model is used to find an (approximate) optimal control policy using, e.g., a variant of Q-learning. We studied a simple example problem and showed how both steps can be carried out using standard learning methods. The methodology delivers a dual controller that beats the maximum likelihood-based certainty equivalence regulator and other heuristic control policies.

Here, we have only studied a scalar example, but the proposed methodology is general and applies to higher-order systems as well. In higher dimensions, a larger number of mixture model parameters will be needed to represent the hyperstate with sufficient accuracy. The actual number will depend on the type of nonlinearities present in the model. For the type of measurement function studied in this paper, leading to a bimodal distribution, a modest number of parameters will be needed, even when the state of the system has a higher dimension. This can be compared to the naive approach of gridding the hyperstate, which requires exponentially more parameters as the system dimension increases. As we explore larger examples, future work will study how the mixture model and the two neural networks and their associated training algorithms can be tuned to the control problem at hand.

9. Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The work was also supported by the ELLIIT Strategic Research Area.

References

A. A. Feldbaum (1960). “Dual control theory. I”. *Avtomat. i Telemekh.* **21**:9, pp. 1240–1249.

Alspach, J. (1972). “Dual control based on approximate a posteriori density functions”. *IEEE Trans. on Automatic Control* **17**:5, pp. 689–693.

Arcari, E., L. Hewing, and M. N. Zeilinger (2020). “An approximate dynamic programming approach for dual stochastic model predictive control”. *IFAC-PapersOnLine* **53**:2. 21st IFAC World Congress, pp. 8105–8111. ISSN: 2405-8963.

Arulampalam, M. S., S. Maskell, N. Gordon, and T. Clapp (2002). “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”. *IEEE Trans. on Signal Processing* **50**:2, pp. 174–188.

Åström, K. J. (1965). “Optimal control of markov processes with incomplete state information i”. *Journal of Mathematical Analysis and Applications* **10**, pp. 174–205.

Bayard, D. S. and A. Schumitzky (2010). "Implicit dual control based on particle filtering and forward dynamic programming". *International journal of adaptive control and signal processing* **24**:3, pp. 155–177.

Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Inc., USA.

Filatov, N. and H. Unbehauen (2000). "Survey of adaptive dual control methods". *Control Theory and Applications, IEE Proceedings* - **147**, pp. 118–128.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The elements of statistical learning*. Springer.

Heirung, T. A. N., B. E. Ydstie, and B. Foss (2017). "Dual adaptive model predictive control". *Automatica* **80**, pp. 340–348. ISSN: 0005-1098.

Klenske, E. D. and P. Hennig (2016). "Dual control for approximate Bayesian reinforcement learning". *The Journal of Machine Learning Research* **17**:1, pp. 4354–4383.

Matni, N., A. Proutiere, A. Rantzer, and S. Tu (2019). "From self-tuning regulators to reinforcement learning and back again". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 3724–3740.

Mesbah, A. (2016). "Stochastic model predictive control: an overview and perspectives for future research". *IEEE Control Systems Magazine* **36**:6, pp. 30–44.

Parsi, A., A. Iannelli, and R. S. Smith (2020). "Active exploration in adaptive model predictive control". In: *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 6186–6191.

Soloperto, R., J. Köhler, M. A. Müller, and F. Allgöwer (2019). "Dual adaptive mpc for output tracking of linear systems". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 1377–1382.

Sorenson, H. W. and D. L. Alspach (1971). "Recursive Bayesian estimation using Gaussian sums". *Automatica* **7**:4, pp. 465–479.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction, 2nd Ed.* MIT Press.

Tse, E. and Y. Bar-Shalom (1973). "An actively adaptive control for linear systems with random parameters via the dual control approach". *IEEE Trans. on Automatic Control* **18**:2, pp. 109–117.

Tse, E., Y. Bar-Shalom, and L. Meier (1973). "Wide-sense adaptive dual control for nonlinear stochastic systems". *IEEE Trans. on Automatic Control* **18**:2, pp. 98–108.

Villani, C. (2009). *Optimal transport: old and new*. Berlin: Springer.

Wittenmark, B. (1995). "Adaptive dual control methods: an overview". *Adaptive Systems in Control and Signal Processing 1995*, pp. 67–72.

Paper III

Model-Free Adaptive MIMO Control of a Chiller Process Using Reinforcement Learning

Christian Rosdahl Bo Bernhardsson Bryan Eisenhower

Abstract

A chiller is a process used to cool down water that is used for cooling of spaces. The overall goal of the chiller control is to achieve the specified temperature sufficiently fast in an as energy efficient way as possible. At optimal operating points, some of the internal variables tend to lie close to some of the many process constraints. Therefore, it is important to find controllers that can reach and remain close to an operating point with as small deviations as possible, to be able to reach good performance without exceeding any constraints. The process has two inputs that nominally are determined by two separate PI controllers using two separate measurement signals. We present an algorithm based on reinforcement learning that can be used to compute a decoupling matrix that combines the two PI controller outputs so that we get a multiple-input multiple-output (MIMO) controller where each control signal depends on several measurement signals. It is demonstrated by simulations on a process model that this method is able to find a controller that performs better than the nominal controller in terms of decreasing the size of deviations from the operating point for some interesting signals while still following reference changes and rejecting disturbances as desired.

Manuscript prepared for journal submission.

1. Introduction

This section provides a brief overview of basic concerns in vapor compression system control. Characteristics of the plant are described, including an overview of the components and the resulting control-oriented behavior that impacts the design. In addition to this, challenges due to large variance in architectures and system variants are outlined. Typical control objectives are described as well as common techniques to achieve these objectives.

A complete overview of challenges and state-of-the-art solutions to vapor compression control is out of the scope of this document, but we hope that this provides a good introduction for further reading.

1.1 Plant Characteristics

Vapor compression based cooling and heating equipment is typically realized as a variant of a refrigeration cycle, which leverages the phase change of a refrigerant (liquid/vapor) to move heat from one environment to another. Fig. 1 illustrates the components of the cycle and its representation in a two-phase pressure-enthalpy diagram for the refrigerant. The area with the dense grid is the two-phase dome, where the refrigerant is two-phase. To the left of the two-phase dome, the refrigerant is a liquid, and to the right of it, it is in gas phase.

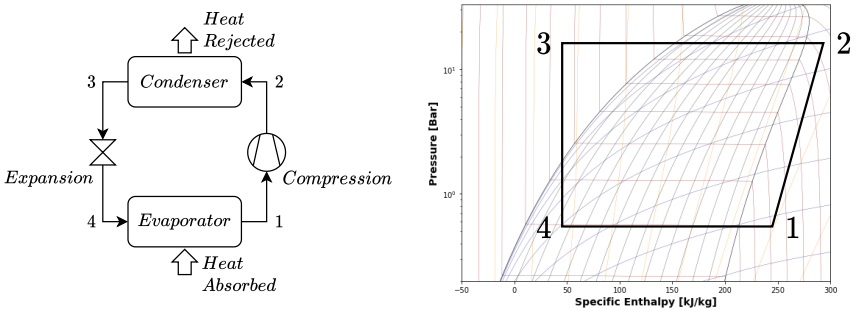


Figure 1. Components of a basic vapor compression cycle (left) and the thermophysical process outlined on a property diagram of the refrigerant that is cycled through the loop (right). The components and process flow are described in the text.

Starting at point 1, the refrigerant enters a compressor in a superheated vapor state (i.e., to the right of the two-phase dome), where the refrigerant is heated through compression, increasing its enthalpy and pressure (to point 2). The fluid then enters the condenser, where its temperature is often much higher than the environment (in many cases, the ambient outdoor temperature) that drives the heat transfer process, cooling the refrigerant and subsequently condensing it (to point 3).

The refrigerant is then rapidly expanded with negligible heat transfer to a low pressure (and therefore low temperature) that is much colder than the other environment that is to be cooled (point 4). This temperature differential drives heat transfer yet again, in this case causing the refrigerant to evaporate to the condition appropriate for compression while cooling down an environment.

Fig. 1 is the most basic refrigeration cycle. There are a number of options for each of the components (fixed speed, variable speed, digital, or multiple compressors; electronically or thermostatically controlled expansion; water or air on the outside of the heat exchangers; etc.). In addition to this, several different architectures have been adopted to increase flexibility and system efficiency (see Section 1.1).

The range of sizes of vapor compression cycles is vast. They can be extremely small for cooling electronics (≈ 100 watts), or they can be many hundreds of kilowatts for providing cooling capacity for an entire building or a group of buildings. The range of applications can also be wide. On the one hand, it is necessary to have a refrigerant within the cycle (a medium that has certain phase-change properties), while outside of this circuit the heat transfer can be exchanged with air, water, or anything else that acts as a heat source or sink (e.g. even conduction to a solid).

There are many aspects of HVAC equipment that present challenges to control design and performance, in no order of importance: component nonlinearity creates variation throughout the operating envelope and introduces challenges during startup and mode switching, the system is spectrally stiff (i.e., there is a large separation between time constants for different aspects of system performance), and the thermofluid system is in closed loop, creating coupling between all components. From a control design and verification perspective, the schematic in Fig. 1 is as simple as it gets, while current products typically have more complicated architectures. In addition to this, a product line typically has a wide range of variants to provide design flexibility to meet different market needs. The control design must account for all of these factors; each is described in more detail below.

Nonlinearity. Every component in a vapor compression system is nonlinear; the compressor and expansion device both operate on pressure–flow relationships that vary at operating conditions, and the heat exchanger performance is fundamentally driven by pressure-flow relationships *as well as* a nonlinear heat transfer coefficient. The nonlinear pressure drop relationships follow a Bernoulli-type quadratic relationship between this drop and flow rate, while the valves often operate in a choked state. The compressor has a flow relationship that is tightly dependent on its design and can exhibit surge instability if the system employs a centrifugal compressor.

The heat transfer behavior varies non-monotonically through the operating range as the relative amount of liquid or vapor changes at various conditions, as they are both functional to the flow rate as well as void fraction of the state of the fluid. That is, heat transfer typically increases with flow rate of the medium, while it is also dependent on the heat capacity (fluid versus vapor), and given the boiling

and condensing process that occurs down the flow path, this relationship becomes complex.

At the systems level, these component nonlinearities are observed as gentle systems level variance in time constants and steady-state gain across the operating envelope. In [Schurt et al., 2010], a H_∞ metric is used to assess clustering of dynamics to regions suited for gain schedule-based control to handle this variation.

Stronger nonlinearity exists in certain regions of the evaporator (or condenser) when its operation takes the state of the fluid to the boundary of the two-phase dome (visible in Fig. 1). In these regions, the state of the fluid approaches pure two-phase vs. single-phase, and the heat transfer and pressure drop behavior changes drastically. This results in a rapid change in gain, and a simple SISO controller used to control these components typically experiences what practitioners colloquially call *hunting* (a control-induced limit cycle). This onset in this instability has been termed *minimum stable superheat* (MSS). [Chen et al., 2002; Shang et al., 2015; Chen et al., 2008]

Abrupt nonlinear behavior is evident during mode transitions that can include, but are not limited to, startup and shutdown, staging (adding or subtracting a compressor), and defrost (redirecting heat to melt ice that may have accumulated on a heat exchanger). From a physical perspective, the nonlinearity occurs from rapid migration of refrigerant charge (mass) and oil throughout the system. From a mathematical perspective, the system goes through a number of singularities as the polygon in Fig. 1 is expanded (or contracted) from a single point. [Li and Alleyne, 2010]

Eigenvalue Spectrum. Vapor compression equipment has a wide variety of timescales that impact modeling and control. These timescales can range a few orders of magnitude, from fractions of a second for flow behaviors, to minutes for larger thermal time constants of the equipment. [Rasmussen and Shenoy, 2012; Gordon and Asada, 2000] On the fastest scale, thermofluid behavior of the refrigerant contains subsecond momentum dynamics that can be relevant to some control objectives, often those related to pressure maintenance. On the other hand, the thermal transients in the system are often governed by not only the refrigerant but also the amount of metal in the heat exchangers and compressor, which can be square millimeters (e.g., for a refrigerator) or square meters (for a whole-building chiller). From an external viewpoint, excitation and disturbances are typically on the order of minutes at the fastest (to react to grid, datacenter, or occupant demands) to months or seasonal to react to slower operating point changes.

For local disturbance rejection and tracking analysis, first-order plus time-delay models are sufficient for control of thermal behavior, while second order behavior is often evident in the pressure response. However, to capture nonlinearity at all operating points and quantify a total system design, it is necessary to assess a much broader range of timescales for comprehensive analysis.

Coupling. Because of the closed-loop thermofluid architecture (Fig. 1), significant coupling exists between all of the components in a vapor compression cycle.

In steady state, the mass flow rate is equivalent for all components, which couples the pressures (and therefore temperatures) of the system. During transients, the driving and restoring forces in the fluid interact because of feedback, and this equivalent flow rate no longer holds as components are filled or drained of refrigerant. From a controls perspective, it has been observed that closing the loop on mass and energy moves poles towards the right half complex plane, slowing the system down. [Morud and Skogestad, 1996]

Historical methods of wrapping SISO controllers around various actuator-component pairs aggravate this coupling even more. As an example, on just a simple 2 input 2 output circuit, it was shown that interactions between actuators (expansion and compression) attenuate single loop gains by a factor of 4.0. [He et al., 1997] Multi-variable control of vapor compression systems is a largely unrealized opportunity (outside of research) and will be discussed later.

Architectures. The simple circuit in Fig. 1 is as basic as it gets. The evolution of market applications and demand, as well as the need for flexibility and higher efficiency, has driven significant complexity into vapor compression system architectures. Fig. 2 illustrates a few examples.

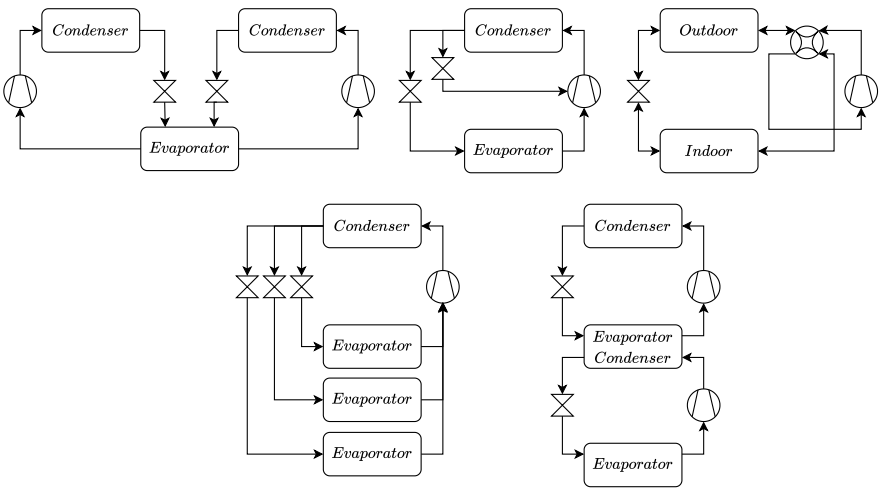


Figure 2. Non-exhaustive examples of different plant architectures (left to right, starting from the upper left): dual circuit, economized, reversible (heat pump), multi-evaporator, and two-stage. A product line may have some of these options as choices, and the controller needs to be quickly configured to the option selected.

These architectures may include

- **Multiple Loops:** To gain flexibility in capacity at higher efficiencies, it is

often beneficial to have two small variable capacity circuits over one larger (but also variable) capacity circuit.

- **Compressor Economizer:** In this circuitry, the expansion process is broken into two stages with some of the refrigerant entering the second stage of the compressor, requiring less lift when compressed. This reduces total refrigerant flow and compressor power consumption for some conditions in the operating envelope, improving system efficiency.

- **Reversible Cycle:** For heatpumps, a series of valves are used to switch the direction of flow through each of the main heat exchangers. The condensation and evaporation are alternated in a given heat exchanger depending on the mode. This application is used to both heat or cool an environment (not at the same time).

- **Multi-Evaporator Systems:** Multiple evaporator systems are used to distribute cooling to multiple locations using the refrigerant circuit (rather than distributing cooling through air or water).

- **Two-Stage Systems:** For applications with a large temperature difference between the two environments, two isolated systems can be used to achieve more flexibility and efficiency. These types of systems are often used in refrigeration settings (e.g. food, medical supplies, etc.).

Although the architecture of a system often does not change once designed, tools and techniques for control design need to be flexible and easily adapted to a variety of architectures. This variety presents a significant challenge to a practicing engineer.

Variants. To maintain a small design and manufacturing base and to balance needs for a large variety of capacity and efficiency, vapor compression products are often designed with a large number of variants. As an example, a water cooled chiller sold by Carrier, designed for an application range of 1,055 to 2,461 kW with a variety of other options for installation specifics or to impact system efficiency, includes the options in Table 1.

Although these options may be correlated (e.g. larger sized heat exchangers typically only pair with larger sized compressor options), the number of variants can exceed many hundred thousand considering the options in Table 1 and others not listed. These are needed to ensure widest market penetration and longevity of the duration of the manufacturing lifetime.

The controller needs to be configurable to all of these variants and offer acceptable performance for any option. This is a significant challenge as it is nearly impossible to test every potential configuration prior to their sale and field commissioning.

Table 1. Example number of options for one chiller line. [Carrier, n.d.]

Component	Number of variants
Refrigerant	2
Evaporator frame	3
Evaporator length	2
Evaporator passes, tube diameter, tube count	12
Voltage supply frequency	4
Motor size	3
Economizer option	2
Condenser frame	3
Condenser length	2
Condenser passes, tube diameter, tube count	8

1.2 Control Objectives

In this section, a high-level description of the problem formulation for control is provided. There are three main goals of the control system: (1) to protect the equipment from self-destruction, (2) to provide sufficient cooling or heating capacity, and (3) to provide this capacity at the lowest energy expenditure possible (highest efficiency).

Component Protection. The compressor in these systems is often known as its *heartbeat*. It is often the most complicated and expensive component, and it provides the propulsion for the refrigerant. Moving parts inside the compressor often require lubrication (oil) that circulates through the circuit in Fig. 1 as a normal process. However, in some cases excessive oil circulation can occur based on controller behavior, including poor control of the degree of evaporator superheat (point 1 in Fig. 1). Loss of this superheat results in the state of the refrigerant entering the two phase dome, resulting in some amount of liquid in its state.

Flooding of liquid refrigerant at the compressor inlet can create problems; many compressors are intended to compress vapor, not liquid, and the additional liquid refrigerant *washes* oil from its surfaces. This creates two issues: its lubrication is necessary for the longevity of the compressor, and excessive oil circulation through the cycle decreases heat transfer between the refrigerant and heat exchanger walls, diminishing performance and efficiency. Therefore, precise control of the inlet of the compressor is essential.

A compressor envelope is typically provided as a steady-state design constraint to ensure compressor health. One objective of the controller is to maintain operation within these boundaries. Fig. 3 illustrates an example protection constraint for a Copeland compressor.

Protections exist for other components. Often, these exceed the boundaries of the compressor protection envelope, yet are activated on different timescales. For instance, high pressure limits may exist for short interval spikes that may rupture

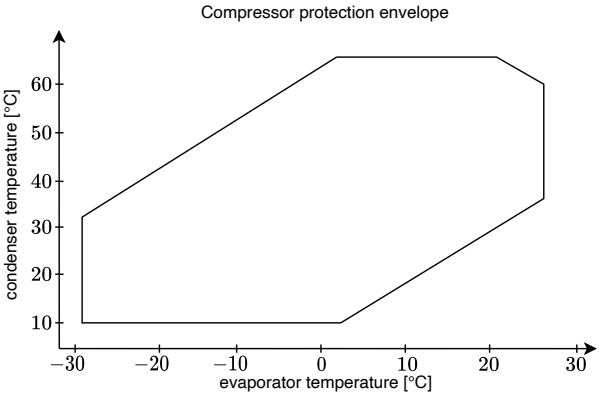


Figure 3. Typical protection envelope for a compressor used in a vapor compression system. [Emerson, n.d.] The control system needs to regulate system variables within this map while meeting the capacity and efficiency targets.

pipes or other components. This protection is needed on a shorter timescale than those driving longer-term degradation (as with the oil management listed above). Another common protection for systems exposed to subfreezing conditions is frost protection.

Capacity Delivery. Capacity is the amount of cooling or heating that a vapor compression system delivers. In most cases, the capacity is delivered through convective heat transfer using a medium (typically air or liquid) that passes through or over the refrigerant piping in the heat exchangers. Vapor compression systems are commonly designed using any combination of liquid or air over the evaporator(s) and condenser(s).

Systems are designed and sold based on rated, nominal, or part load capacity descriptors. These values describe system performance under industry-standard conditions (much like fuel efficiency in an automobile). The conditions often include a small number of steady-state values and at times a dynamic or cycling test that are aggregated as a single performance metric. [CEN, 2018; AHRI, 2017]

The objective of the control system is to meet the published conditions robustly and provide sufficient performance off of these conditions. Unfortunately, much like in the automotive industry, system performance is not as published in the real world. In [O'Hegarty et al., 2022], actual tests indicated 40% lower than the rating value performance in the field due to a number of reasons, including poor test definition (not enough climate zones or assumptions on typical humidity), neglect of some heat losses in actual products, and underestimating the effects of on-off cycling. With more attention on remote monitoring and connectivity, actual observed and measured controlled performance is becoming more relevant as this becomes more

widely measured.

Optimal Efficiency. Providing adequate capacity is of primary importance. However, in vapor compression systems, it is possible to achieve this capacity in multiple ways. Heat transfer is driven by a temperature difference in a closed environment. In an open environment with flow, convective heat transfer is enhanced by the flow of media as well. Due to the variety of actuators, and the bilinear nature of convective heat transfer (flow multiplied by a temperature potential), and because of the nonlinear behavior of thermofluid flows, there is often flexibility in steady-state operation (setpoints) that provide safe and optimized operation – high flow with low temperature potential can provide the same capacity as lower flow and higher temperature potential.

The steady cooling capacity for the system in Fig. 1 is calculated as the flow of the refrigerant in the circuit times the enthalpy difference (point 1 minus point 2). Given the constraint that point 1 shall not enter the dome, system efficiency can be altered (for the same capacity) by adjusting the degree of evaporator superheat (distance between the dome and point 1) as well as point 3 (subcooling at the condenser outlet).

At the component level, evaporation efficiency is enhanced by the turbulent activity of bubble formation as well as the presence of liquid (higher heat capacity). For the same flow rate, the vapor has a much smaller heat exchange gain, and therefore it is beneficial to maintain point 1 as close as possible to the two-phase dome. On the other hand, refrigerant charge is conserved in the circuit and what is present in the evaporator must be taken from the condenser. The condenser has similar efficiency behavior as a component, and therefore systems-level efficiency is critical.

Several techniques are used in practice to maximize system efficiency without excessively complicating the control system. These approaches typically involve setpoint maps or extremum seeking that set or seek system efficiency at different operating conditions. These approaches are discussed below.

1.3 Control Technologies

Control systems for vapor compression cycles have evolved from purely on–off and mechanical to the gradual use of more electronically controlled components (compressors, valves, pumps, fans) in most applications. Unlike in building systems control, where controls technology is either suggested by standards [ASHRAE, 2021], or described in textbooks [Mitchell and Braun, 2012], local equipment control techniques are often proprietary to the manufacturer.

A review of control technologies with a focus on control-oriented modeling was performed in [Goyal et al., 2019]. A small number of examples are referenced in conventional, high performance (e.g. MPC, MIMO), and intelligent techniques (machine learning, fuzzy logic). An analog is provided in [Afram and Janabi-Sharifi, 2014] with the scope expanded to building energy systems (not onboard equipment

control). Given the brevity of literature on this topic, a brief review of technologies used in vapor compression control is provided below.

A generic block diagram of a plant (cycle in Fig. 1) is presented in Fig. 4. The components of the control system include (but are not limited to) the ones listed in Table 2, and the variables are described in Table 3.

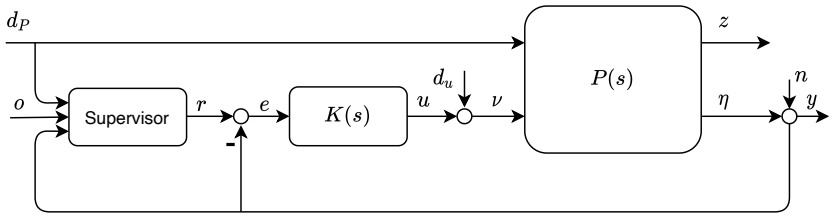


Figure 4. Typical closed-loop system on the plant $P(s)$ using the regulator $K(s)$ and setpoint specifier (Supervisor). The variables are described in Table 3.

Table 2. Components of the generic control system.

Supervisor	A decision maker that specifies setpoints for the controller to track based on disturbances, the objective of the system, and feedback (often slow). The supervisor may include logic for protections and overrides (which would select different outputs and references to track with the controller $K(s)$).
$K(s)$	This regulator contains elementary control functions like PI controllers.
$P(s)$	The plant or system to be controlled.

Variables that are communicated between the components in the block diagram may include:

State Machines. State machines are typically employed in vapor compression control systems as a way to manage the control objectives listed above. The state machine may set or infer the operating state, or perform lower level management of setpoint tables or status of on-off devices. An obvious choice of states of the system is the operating modes, whether it be the startup or shutdown state (an example is described in [Pršić and Vičovac, 2017]) as well as protection, overrides, or other operating objectives.

Although the authors are unaware of the use of formal methods to assess stability and performance of the resulting hybrid control systems for all protection and mode handling, there has been work to understand subsets of these functions, including on–off commands. [Zhang et al., 2015; Li et al., 2012] The study in

Table 3. Variables in the control system.

o	Objective. This is the high level objective of the machine (e.g. cooling vs. heating, defrost or normal operation, free cooling mode).
d_P	Ambient conditions and disturbances. Outdoor conditions, flow rates (specified outside of the system). These disturbances take the system to different operating points (nonlinearity) and may disturb the dynamics as well (small signal disturbance).
v	Actuator input(s). Valve, compressor, and fan or pump commands.
z	Performance variables. Conditions of the system that are important but not fed back to the controller (difficult to measure), but may be used in design or analysis.
y	Process feedback. Variables that are used in the regulator. These variables may change based on conditions, objectives, or overrides.
r	Reference(s). Tracking variables.
e	Error. Difference between the reference and system output.
u	Controller output. Commands sent to actuators including valve opening, speed, or on-off state as examples.
d_u, n	Disturbances. Input and output disturbances or noise.

[Uhlmann and Bertsch, 2012] quantified the efficiency loss due to cycling a heat-pump on and off to regulate its capacity.

Among the issues related to state machine stability and performance are design tools and strategies (considering the variants and architectures listed in Section 1.1), as well as effective sensing strategies. In many cases, it is unrealistic to measure all of the states that drive transitions, leading to expensive or unique sensing (e.g. imaging for frost formation as in [Zheng et al., 2019] or [Xiao et al., 2009]).

Regulation and Disturbance Protection. The *inner* loops of a vapor compression system controller provide actuator commands to the compressor, valves, and fans or pumps in the system. Disturbance rejection is often the primary concern, as these systems are rarely designed to track complicated setpoint reference trajectories. However, in some cases supervisory control re-defines setpoints as a means to respond to demand or to optimize performance.

Like many process systems, one of the most popular control techniques for vapor compression system regulators is PI control – although given the evolution towards more complicated architectures and variable actuation, limits are quickly being realized in these traditional approaches.

Among many challenges is component and system nonlinearity (Section 1.1) driven by the need for compressor protection. The evaporator leaving superheat

(which enters the compressor) is of key concern; this is often actuated by the expansion valve. The evaporation process and the input-output relationship are nonlinear, which often results in suboptimal performance, typically observed as control-induced oscillations. Gain scheduling can help with this, but given the number of variants, architectures, and variance across the operating envelope, it is a challenge to guarantee that this works well. Nonlinear control techniques have proven to be beneficial for dealing with this nonlinearity.

In [Rasmussen et al., 2006] it was shown through reduced order modeling that if the moving length of the superheated region is used as an output variable (traditionally: superheat temperature), and the evaporator mass flow as input variable (traditionally: expansion valve command), a linear system is formed – which significantly simplifies the PI design approach. Similar work was done in [Elliott and Rasmussen, 2010], where a cascaded algorithm was used to reduce the nonlinearity to avoid gain scheduling. Nonlinear PI analysis and control using describing function methods was performed in [Rehrl et al., 2009], while the scope of the plant was HVAC-to-building integration rather than onboard equipment control.

In [Vinther et al., 2013], extremum seeking is used to attempt to find the inflection (slope change) of the nonlinear evaporation phenomena, resulting in a measurement-based approach for predicting the onset of instability (i.e., the smallest stable superheat for a certain operating condition). Similarly, in [Jolly et al., 2000], MSS line theory is used as a primary design element for stable superheat control.

As mentioned in Section 1.1, the coupling in the system is a source of challenges and an avenue for closed-loop performance improvement. The seminal work in [He et al., 1997] outlined a MIMO-based control strategy, and several research groups have maintained activity in this area over the years. In [Anderson et al., 2002], MIMO control design of a 3×6 system provided 3–5 times improvement in transient response, and in [Shah et al., 2004], a detailed workflow is given to design estimation, multivariable control, and adaptation for a vapor compression system. The benefit of a MIMO LQG controller over a set of SISO-PI controllers was demonstrated in [Jackson et al., 2019].

Common to all of these studies are detailed model-based analysis and single-variant designs, which in many cases are proven or demonstrated in a modeling environment or using an isolated experiment. The hardening and scaling of the design process for industrial implementation remains a challenge.

Supervisory and Optimization. Supervisory control can be used to select between different active control regulators, constraints, or setpoints based on the current operating conditions. A common approach to handle this logic is through operating condition maps – lookup tables or low-order polynomials that provide setpoints or logic based on current operating conditions. These algorithms are often designed using models, experimental results, component manufacturer data, or past experience. As this is very cumbersome, and because all variants of a system cannot be tested,

there are efforts to provide online optimization and updates to such algorithms.

System efficiency is predominately driven by compressor operation, as it is often the largest power consumer. The intention is to compress the refrigerant *just enough and no more* than necessary to provide the capacity requested. Heat exchanger efficiency also plays a role in system efficiency. Filling the heat exchangers appropriately with two-phase refrigerant typically leads to optimal performance. This is often measured by the degree of subcooling for the condenser (as optimized in [Yang and Yeh, 2015]), and evaporator superheat.

Due to the large variance in both steady-state performance (i.e. where *optimal* is) and dynamics, online setpoint optimization has been suggested as a solution to provide real-time optimization of system setpoints or controls. Multivariable extremum seeking was performed in [Dong et al., 2015] on a mixture of inner and outer loop control variables (inner loop – fan speeds, outer loop – superheat setpoint), resulting in energy savings of 7.6% in a model-based evaluation. Optimal subcooling through extremum seeking illustrated 9% performance improvement [Koeln and Alleyne, 2014], and a number of other studies have been performed to address the issues of time scale overlap that can make implementation of such approaches in the field impractical. [Burns et al., 2018; Dong et al., 2018]

Fault Handling and Diagnostics. Vapor compression systems and their components can experience sudden failure or long-term degradation, observed as off-design performance or frequent overrides and shutdown. Unlike personal consumer products, the installation and commissioning of these systems is non-exact, and poor performance can be experienced even when they are new because of installation error. If we consider faulty operation as *malfunctioning or anomalous deviations from expected behavior*, the list of reasons is large, including mechanical, software, or procedural sources of error.

Typical fault sources include poor performance of heat exchangers, compressor(s), or valves, as well as issues with electronics or installation. Controls is often a leading cause of poor performance. In [Breuker and Braun, 1998], control and electronics were the source of 40% of faulty behavior (controls was identified as a leading cause in [Madani and Roccatello, 2014] as well). Most of the published work in fault analysis of vapor compression systems is in identifying key faults, modeling faulty behavior, and developing monitors and diagnostic tools.

Quantifying, ranking, and subsequently generating diagnostic algorithms for faults has been performed to identify which faults matter. A number of single-variant experiments have been performed that quantify and rank the impact of mechanical faults on system performance. [Kim et al., 2006] The amount of refrigerant charge in the circuit is crucial to system performance. Since the system is closed-loop, the charge is conserved, yet leakage is a significant concern. [Choi and Kim, 2002; Goswami et al., 2001] It is unrealistic in almost all cases to maintain a measurement of the mass of the system (and therefore the mass of charge in the system), and so a number of approaches are used to infer the charge quantity from common

sensor readings (typically pressure and temperatures).

Vapor compression system models (both steady-state and transient) are often used for component and control system design. Off-design conditions are often part of the testing and development process, while capturing catastrophic failure is often out of the scope of most studies. In most cases, model-based analysis takes physical component parameters to their limit of validity to analyze the resulting performance. As an example, models for non-standard charging, heat exchanger fouling, compressor valve leakage, and liquid line restriction can be found in [Cheung and Braun, 2013].

For online diagnostics, the industrial focus has been in further connectivity (over the air control updates, remote monitoring and data collection), while fault tolerant control is in its infancy. A recent review of the status of diagnostics in HVAC systems can be found in [Bellanco et al., 2021]. Because of the coupling in the thermofluid circuit, and moreover feedback controls, the observability of system states contributing to faulty behavior as well as identifiability of parameters leading to a fault may not be possible with a low-count sensor set. In [Li and Braun, 2007], *virtual sensors* generated from first principles were assessed based on their ability to decouple different faults, and [Lu et al., 2022] found a benefit of well derived algorithms over excessive sensor sets.

Charge loss is one of the most frequently observed faults that garners significant attention. [Tassou and Grace, 2005] provides an example of charge loss diagnostics (steady state), while dynamic considerations in [Yun and Chang, 2021] found that charge loss can be identified by the dynamic response of the condensation temperature and degree of subcooling during system startup in laboratory conditions.

2. Problem Formulation

In this report, we consider the problem of constructing an adaptive multiple-input multiple-output (MIMO) controller for control of a chiller process. The main components of the process in question are illustrated in Fig. 5. The purpose of the process is to cool down water that in turn is used for cooling of some given space, e.g. a room in a building. This is achieved by letting the water circulate between the space to be cooled and the evaporator. The primary goal of the chiller process is to decrease the temperature of this water so that it attains some specified value when leaving the evaporator. As a secondary goal, we would like to achieve this in a way that is as energy efficient as possible. Our aim is thus to find a controller that achieves both these goals, within the limits and constraints of the process.

Heat that is extracted from the evaporator water is transferred to a separate water circulation, through the condenser, where it is transported away to the outdoor air. Heat transfer is achieved by means of a refrigerant that circulates between the evaporator and condenser, while undergoing pressure changes. The refrigerant cycle is illustrated as the black curve in the *ph* diagram in Fig. 6, where p is the pressure

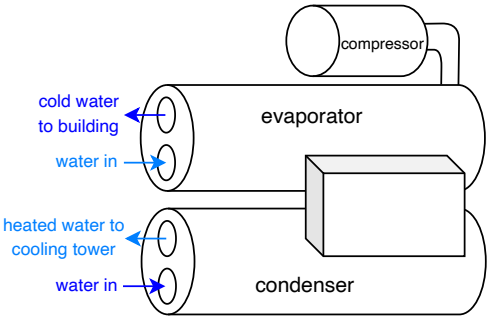


Figure 5. The main components of the chiller process are an evaporator that cools down water used for cooling down a building, and a condenser that extracts the heat using a secondary water circulation. The heat transfer is carried out by means of a refrigerant, of which the pressure is varied using a compressor and an expansion valve.

and h is the specific enthalpy (enthalpy per unit mass) of the refrigerant. The blue curve shows the border for where some phase transition starts, and the orange lines are some level curves for the temperature. The pressure is changed by means of the compressor and an expansion valve. These are our two actuators, and thus the control signals will be the compressor frequency and the degree of opening of the expansion valve.

The primary measurement signal used for feedback control is the evaporator leaving water temperature (eLWT). As for the choice of a secondary measurement signal, there are several options. One option is the level of refrigerant in the evaporator. For efficient cooling performance, the refrigerant should almost cover the water pipes in the evaporator. However, if the level is too high, the performance will decrease. Hence, it is assumed that making sure that the refrigerant level is always close to covering the water pipes is a good aim for efficient operation of the process. In practice, this level can usually not be measured. Therefore, it is interesting to also consider other signals that could provide some information about the performance but that are measurable. One such signal is the subcooling, which is the difference between the temperature of the refrigerant when leaving the condenser (point 3 in Fig. 6) and its boiling temperature. In this report, we will use the eLWT and the subcooling as feedback signals for our controller.

The controller should be able to achieve both good reference following for the two feedback signals and good disturbance rejection. Some disturbances that can occur are, e.g., variations in temperature or flow in the water passing through the evaporator or the condenser. Since variations in the condenser water tend to have a smaller effect on other interesting signals, we will focus on variations in the evaporator water.

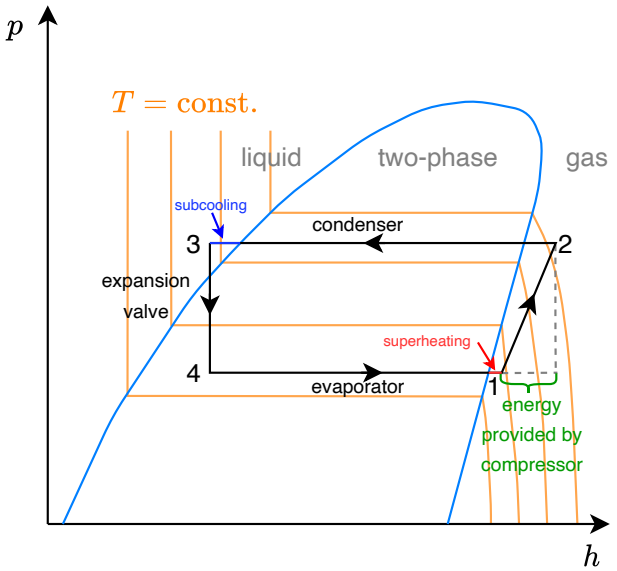


Figure 6. The refrigerant cycle for the chiller process. The black curve shows the pressure p and the enthalpy h of the refrigerant at each stage of the refrigerant cycle. The blue curve shows phase transitions, and the orange curves are level curves for the temperature.

In conclusion, the system model that we consider will have

- **control signals:**
  - $u_1 =$ expansion valve opening ($u_1 \in [0,1]$),
  - $u_2 =$ compressor speed ($u_2 \in [0,210]$),
- **feedback signals:**
  - $y_1 =$ subcooling,
  - $y_2 =$ evaporator leaving water temperature (eLWT),
- **disturbances:**
  - $d_1 =$ evaporator entering water temperature (eEWT),
  - $d_2 =$ evaporator water flow (eFlow).

In the controller structures that are common for controlling this type of processes in practice today, the compressor speed u_2 is only determined by feedback from the eLWT y_2, while the expansion valve opening u_1 is only determined by

feedback from the other feedback signal y_1, which in our example is the subcooling. The purpose of our study is to investigate whether using a combination of several measurement signals to determine each control signal can lead to better control performance compared to this nominal case.

In practical operation, the system has many constraints on different signals that must be satisfied. An example of this is that the pressure of the refrigerant when entering and leaving the compressor must stay within certain bounds. Another example is that the heating of the refrigerant in the evaporator must be large enough to ensure that the refrigerant is completely in gas phase when entering the compressor. Not satisfying this might damage the equipment. The most energy efficient operation is probable to occur at points when some signals are close to some of the constraints. It is therefore important to construct a controller that keeps the maximum deviations from the nominal value of these signals as small as possible. This allows one to choose setpoints such that the efficiency is large without exceeding the constraints.

Besides constructing a controller that minimizes variations in relevant signals induced by disturbances and reference changes, we would also like the controller to be adaptive. Since the process in practice is adapted to each customer and circumstance, there is a huge number of variants of the process. For this reason, we would like to have a controller structure that has the ability to adapt to different process variants. The adaptivity could also allow for the possibility to adjust the controller for a specific process with time when some process parameters change slightly. If we have a sufficiently accurate known process model, we could use model-based control, using the model with known or estimated parameters to construct the controller. Another approach is, however, to use model-free control, where the controller adapts to the current process without needing an accurate model. Our main goal will be to present an algorithm for constructing a model-free controller. However, we will compare the performance of a controller obtained by this method with a model-based controller on a simulated example process.

3. Preliminaries

3.1 Static and Dynamic Input Decoupling

A simple control strategy for a system P with two inputs and two outputs is to use two separate PI or PID controllers to control it. This decentralized PI control architecture is illustrated in Fig. 7. The process P is divided into two interacting blocks, P_1 and P_2, each containing one input and one output. In this solution, we have two PI controllers, C_1 and C_2, which are responsible for one of these blocks each. Thus, C_1 only uses output y_1 to determine input u_1, and analogously for C_2, y_2 and u_2. The decentralized control structure is easy to understand intuitively. The main drawback is that there will be strong interactions between the two control loops, and to maintain a closed-loop stable system one might have to detune the

closed-loop bandwidth, resulting in a quite slow system, increasing the problem with disturbance attenuation.

In order to decrease the problem with interactions between different parts of the process, we can use a method called *input decoupling*. [Skogestad and Postlethwaite, 2005] The structure of the method is shown in Fig. 8. Here, the decoupling block

$$D = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}$$

can consist of either constant gains (static decoupling), or dynamic transfer functions (dynamic decoupling). We can view the structure as that we use both outputs, y_1 and y_2, for determining each of the inputs u_1 and u_2. Alternatively, we can view the decoupling matrix as a part of a modified process $\widetilde{P}(s)$ that has two inputs $\widetilde{u}_1$ and $\widetilde{u}_2$, which are determined by two separate PI controllers, and which define the inputs to the original process $P(s)$ according to

$$\begin{aligned} u_1 &= D_{11}\widetilde{u}_1 + D_{12}\widetilde{u}_2, \\ u_2 &= D_{21}\widetilde{u}_1 + D_{22}\widetilde{u}_2. \end{aligned}$$

Static Input Decoupling. The most common choice is to use static decoupling and first choose D to achieve perfect decoupling in stationarity, i.e.,

$$P(0)D = I, \tag{1}$$

and then apply decentralized controller design for the transformed system

$$\widetilde{P}(s) := P(s)D.$$

For the chiller process, this could for example mean that the control system consists of two parts, where

> PI controller C_1: Measures and maintains the desired condenser subcooling by manipulating the control signal combination $\tilde{u}_1$,
>
> PI controller C_2: Measures and maintains the desired evaporator leaving water temperature, eLWT, by manipulating the control signal combination $\tilde{u}_2$,

see Fig. 8.

An interesting alternative to the conventional input decoupling is so-called inverted decoupling, illustrated in Fig. 9. This architecture is claimed to have some advantages in [Hägglund, 2012], specifically when it comes to the design of an anti-windup mechanism, responsible for handling situations of controller saturation.

If the decoupling equation $P(0)D = I$ is satisfied, then the two control loops are perfectly separated, in stationarity. This means, for example, that a setpoint change in the reference level for y_2 – for instance corresponding to the evaporator leaving water temperature – would not result in any stationary change in y_1, which could be the condenser subcooling. This would be an advantageous property for optimal energy-efficient operation of the chiller.

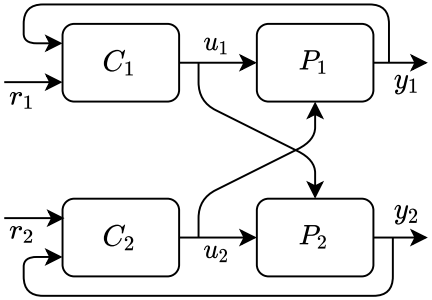


Figure 7. Decentralized PI control. A problem with this architecture is that the interactions between the two loops can lead to reduced closed-loop performance, since it is often hard to achieve a fast closed-loop system.

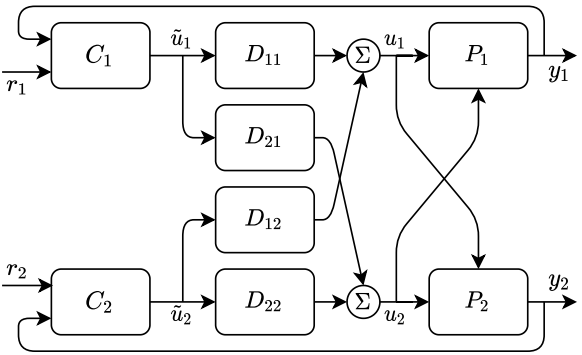


Figure 8. Control by conventional decoupling. The elements in the decoupling matrix D are chosen to minimize the interaction between the control loops for variables y_1 and y_2. These elements can either be static gains, or dynamic elements. The decoupling architecture typically allows for higher controller loop bandwidth and better control performance. It might be a challenge to find a D that is suitable for the entire range of operation of the chiller system.

Dynamic Decoupling. The stationary decoupling equation $P(0)D = I$ results in a controller that achieves decoupling in stationarity but will still give transient interactions between the two loops. Often these transients have only marginal impact on performance and is something that can be accepted. But in situations where these transient interactions need to be reduced, instead a dynamic design equation of the form $P(s)D(s) = I$ should be solved.

In practice, the resulting solution $D(s) = P^{-1}(s)$ is however not possible to implement, since it would result in infinite gain at high frequencies. Therefore, only

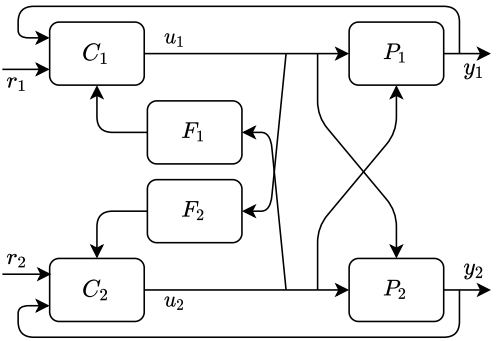


Figure 9. Control by inverted decoupling as described in [Hägglund, 2012].

approximate dynamic decoupling is targeted, where a design equation such as the following can be used to find $D(s)$

$$\min_{D(s)} \|W(s)(P(s)D(s) - I)\|,$$

where $W(s)$ describes a frequency weight and where a suitable system norm, such as H_2 or H_∞, is used. The weighting $W(s)$, usually chosen to be in diagonal form, is a design variable. The dynamic decoupling design is more complicated and will require more modeling knowledge, whereas static decoupling only requires information about the static gain of the system.

A simple alternative to the optimization-driven design described above is to use the dynamic decoupling

$$D(s) = (P(s))^{-1}F(s),$$

where $F(s)$ is a low-pass filter with suitable bandwidth and sufficiently high roll-off, for instance of the form

$$F(s) = \frac{1}{(1+sT_f)^n}I,$$

where the time constant T_f and filter order n are properly chosen.

3.2 Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning that can be useful for finding a control policy without needing a model of the system to be controlled. The basic setup is illustrated in Fig. 10. We have an environment that we can affect by applying some action A_t at each discrete time step t. We assume that the environment can be described as a Markov decision process. This means that, at time t, it can be described by a state variable S_t. Given the state S_t and the action A_t, the environment will transition at the next time point into a new state S_{t+1}. The new state

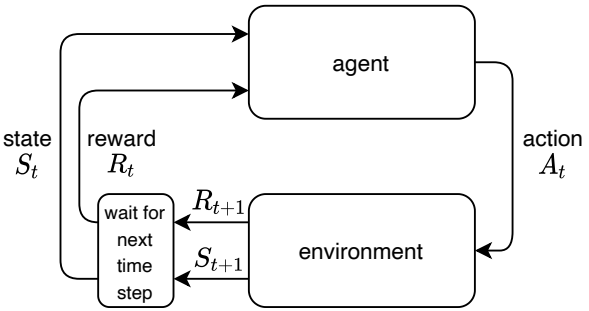


Figure 10. Reinforcement learning.

is given by a random variable that is determined by the current state S_t and action A_t, but does not depend on any previous states or actions. This is called the Markov property. After the state transition, we obtain a reward R_{t+1} from the environment. This is a number that depends on the previous state S_t and the action A_t and is a measure of how good the immediate effect of this state-action combination was. The reward is, in general, also a stochastic variable, meaning that different rewards can be obtained for the same state-action combination when it is repeated.

In the RL setup in Fig. 10, we have illustrated the environment that is affected by an action A_t and which outputs a new state S_{t+1} and a reward R_{t+1}. The second component of the setup is the agent, which is our RL algorithm. The agent's task is to decide which action A_t to apply to the environment. To do this, it can use information about all previous states, actions and rewards. The goal of the agent is to select actions such that the expected value of some weighted sum of future rewards is maximized. This weighted sum is often chosen as

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k},$$

where $\gamma \in [0,1)$ is called the discount factor and G_t is called the return. In this report, we will use the simple case of letting the return be equal to the immediate reward, i.e., $G_t = R_t$. This corresponds to a discount factor $\gamma = 0$. For our purpose, this seems to work well enough without complicating the algorithm unnecessarily.

In order for the agent to decide which action to choose, it can use a so-called value function $V(S_t)$ that gives an estimate of the expected return G_t when being in state S_t and choosing actions according to some given policy. Assume that we know the transition probabilities $P(S_{t+1} \mid S_t, A_t)$ for each possible new state S_{t+1} given that each possible action A_t is applied from the current state S_t. If the value function is a good representation of the expected return for an optimal policy, we can then choose the optimal action as

$$A_t = \arg\max_A \sum_{S_{t+1}} P(S_{t+1} \mid S_t, A) V(S_{t+1}). \tag{2}$$

However, if the value function is not a good enough representation of the expected return, more exploration is needed, in terms of trying out more different actions. This generates more data about how different actions affect expected returns, which can be used to improve the value function $V(S_t)$, so that it gives a more accurate representation. The algorithm should thus find a balance between exploration, to improve the value function, and exploitation, in the sense that we exploit the value function to select an action according to (2).

A common and rather simple method of balancing exploration and exploitation in RL is the ε-greedy method. In this method, we use a variable $\varepsilon \in [0,1]$ that defines the probability of exploration. At each time step, a random action from the action set is chosen with probability ε, while an action is chosen according to (2) with probability $1-\varepsilon$. The variable ε is usually decreased with time, so that we get a high degree of exploration initially, when the value function is a less good estimator of the expected return, and a higher degree of exploitation at later times, when the value function is a better estimator and is more likely to lead to an action that is close to optimal. More details about different RL algorithms and RL in general can be found in e.g. [Sutton and Barto, 2018].

4. Method Description

4.1 Method Structure

Our goal is to investigate whether using a combination of several measurement signals to determine each control signal can improve control performance compared to a nominal decentralized control structure. We will investigate two different methods for achieving this and compare the results with the nominal case. The first method is ordinary model-based static decoupling, where the decoupling matrix D is computed according to (1). The second one, which is the focus of this report, is also a static decoupling, but where the decoupling matrix is determined by a reinforcement learning algorithm instead of being explicitly computed based on a process model.

The setup we will use is illustrated in Fig. 11. The chiller process $P(s)$ is affected by the two control signals u_1 and u_2, and disturbances d_1 and d_2. As outputs from the process, we have the feedback signals y_1 and y_2, and some other measurement signals y_i for $i = 3,\ldots,m$. The reference signals r_1 and r_2 define the desired values of the feedback signals y_1 and y_2. The controller consists of two PI controllers $C_1(s)$ and $C_2(s)$ whose outputs are combined by a static decoupling matrix D, and then used as inputs for the process $P(s)$. Some reasons for using a PI controller based control structure are that this limits the number of degrees of freedom, which should make it easier to find a well-working controller, and that this results in a transparent and well-known structure that can be examined easily.

The nominal case corresponds to the decoupling matrix D being equal to the identity matrix, i.e., $D_{11} = D_{22} = 1$ and $D_{12} = D_{21} = 0$. We will assume that we

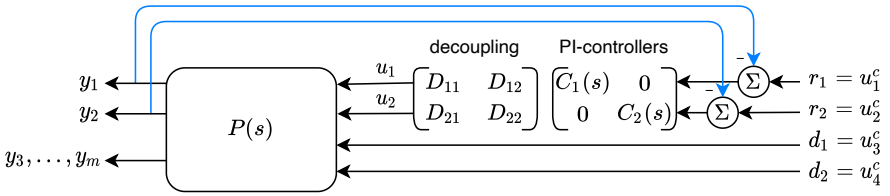


Figure 11. Setup for the closed-loop system. The chiller process $P(s)$ is controlled using the inputs u_1 and u_2, and has outputs $y_1, \ldots, y_m$. The first two outputs y_1 and y_2 are used as feedback signals for the controller to determine the control signals. These should follow given references r_1 and r_2. Moreover, the process is affected by disturbances d_1 and d_2. The references and disturbances are viewed as closed-loop inputs u^c. Each output y_i should be as little affected by each closed-loop input u_i^c as possible, except for that y_1 and y_2 should follow r_1 and r_2, respectively.

know some PI controllers $C_1(s)$ and $C_2(s)$ that have been tuned to give reasonable (but not optimal) performance for this case. The problem we will consider is then whether the four elements of the decoupling matrix D can be chosen such that the control performance is improved compared to the nominal case, while keeping the PI controllers fixed. The control performance is measured in terms of a cost function that depends on the control and measurement signals of the system $P(s)$ obtained when making some reference changes and applying some disturbances. The cost should make sure that the tracking of the reference signals is good enough while minimizing problematic variations in some relevant measurement signals, by applying well-behaved control signals within some specified bounds.

4.2 Goal of the Algorithm

As seen in Fig. 11, our controller K is in the frequency domain described by

$$\begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} = \underbrace{\begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} C_1(s) & 0 \\ 0 & C_2(s) \end{bmatrix}}_{K} \begin{bmatrix} E_1(s) \\ E_2(s) \end{bmatrix},$$

where $U_i(s)$ is the Laplace transform of input $u_i(t)$, and $E_i(s)$ is the Laplace transform of the control error $e_i(t) = r_i(t) - y_i(t)$. The decoupling matrix D is parameterized as

$$D = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} = \begin{bmatrix} S_1^1 & 0 \\ 0 & S_2^1 \end{bmatrix} \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix} \begin{bmatrix} S_1^2 & 0 \\ 0 & S_2^2 \end{bmatrix} = S^1 \widehat{D} S^2, \tag{3}$$

where the matrix $\widehat{D}$ contains the four parameters θ_1, θ_2, θ_3 and θ_4 to be determined, and S^1 and S^2 are predefined diagonal scaling matrices. The purpose of the scaling matrices is to make sure that the parameters to be determined have the same order

of magnitude. This is supposed to make it easier for the algorithm to find a good solution. For given scalings S^1 and S^2, and given PI controllers $C_1(s)$ and $C_2(s)$, the complete controller K is thus defined by the parameter vector $\theta = (\theta_1, \theta_2, \theta_3, \theta_3)^{\mathrm{T}}$, i.e., $K = K(\theta)$, and the goal of our algorithm is to find a good value of θ.

4.3 Cost

To evaluate the control performance for a given parameter vector θ, we sequentially apply a test input signal to each of the inputs of the closed-loop system, which are the reference signals r_1 and r_2, as well as the disturbance signals d_1 and d_2. The test input signals that we use consist of taking one step up and back, and then one step down and back, as shown in Fig. 12. The step magnitudes are different for the different signals. We denote the vector of closed-loop inputs by $u^c = (r_1, r_2, d_1, d_2)^{\mathrm{T}}$. The vector of the resulting measurement signals is denoted by $y = (y_1, \ldots, y_m)^{\mathrm{T}}$. The part of the measurement signal y_j that corresponds to the time range $[t_i^0, t_i^f]$ where a test input as in Fig. 12 is applied to the closed-loop input u_i^c is denoted by y_{ij}. The total time of a test run, where all test inputs are applied for a controller with a certain decoupling matrix, is denoted by T.

After a test run, we evaluate a cost for the obtained measurement and control signals y and u. The cost that we use is

$$\mathrm{cost}(y,u) = \sum_{i=1}^{4} \sum_{j \in \mathcal{J}_i} w_j \max_t |y_{ij}(t)| + \alpha A(y) + \beta B(u) + \gamma C(u),$$

where $\mathcal{J}_i = \{j = 1, \ldots, m : j > 2 \text{ or } j \neq i\}$,

$$A(y) = \sum_{i=1}^{m} \frac{1}{t_i^f - t_i^0} \int_{t=t_i^0}^{t_i^f} w_i \left(\max\{0, y_i(t) - y_i^{\max}(t)\} + \max\{0, y_i^{\min}(t) - y_i(t)\} \right) \mathrm{d}t,$$

$$B(u) = \sum_{i=1}^{2} \max_t \left([w_i^u \max\{0, u_i(t) - u_i^{\max}\}]^2 + [w_i^u \max\{0, u_i^{\min} - u_i(t)\}]^2 \right),$$

$$C(u) = \sum_{i=1}^{2} \frac{1}{T} \int_{t=0}^{T} w_i^u |u_i'(t)| \mathrm{d}t.$$

The first term of the cost is designed to keep the deviations in all measurement signals as small as possible, except for y_i when u_i^c is a reference signal, when changing the closed-loop input u_i^c for $i = 1, ..., 4$. The different measurement signals are weighted by the non-negative output weights w_j for $j = 1, ..., m$. Optimally, input $u_1^c = r_1$ should only affect output y_1, input $u_2^c = r_2$ should only affect output y_2, and otherwise the inputs u_i^c should not affect any other measurement signals. The term $A(y)$ is the cost of tracking errors. The measurement signal should be close to the corresponding reference signal, and we specify how close we would like it to be by two curves $y_{\min}$ and $y_{\max}$, as exemplified in Fig. 13. When the signal is

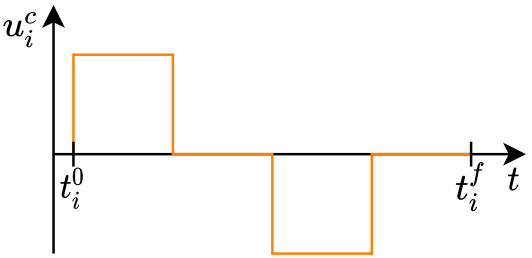


Figure 12. A test input of this type is applied sequentially to each closed-loop input u_i^c in order to evaluate the performance of the controller for a certain parameter vector θ.

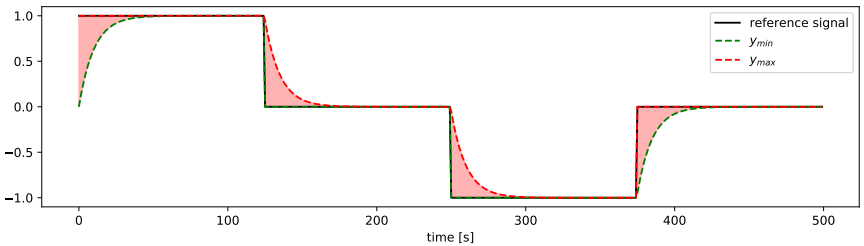


Figure 13. Tracking specification example. We want the measurement signal y to track a reference signal (black curve). When the measurement signal is between the specification curves $y_{\min}$ (green dashed) and $y_{\max}$ (red dashed), no cost is added to the cost function. When the measurement signal is outside of this area, a cost proportional to the deviations is added.

between these curves, no cost is added; and otherwise, we add a cost proportional to the deviation from the area. The next term, $B(u)$, is zero when each control signal u_i remains within some specified range $[u_i^{\min}, u_i^{\max}]$, and gives a quadratic penalty for the largest deviation from each endpoint of this interval. The non-negative coefficients w_i^u are control signal weights. Finally, the term $C(u)$ is added to avoid solutions where the control signal has many rapid variations. The relative weights of the cost terms are set by the constant non-negative coefficients α, β and γ.

4.4 Algorithm

To find good parameter values, we use a reinforcement learning algorithm to train a neural network that approximates a value function $V(\theta)$ that predicts the value of the reward of a simulation with a decoupling matrix defined by the parameter values $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)^{\mathrm{T}}$. The vector θ is the reinforcement learning state. Each parameter is assumed to lie within some known range $\theta_i \in [\theta_i^{\min}, \theta_i^{\max}]$, so that the

state set $\mathcal{S}$ is given by

$$\mathcal{S} = \{\theta : \theta_i \in [\theta_i^{\min}, \theta_i^{\max}] \text{ for } i = 1,2,3,4\}.$$

The reward R of one test run is normally the cost multiplied by -1. However, if some rewards are much smaller than those that are close to the maximum reward, difficulties can occur in accurately representing the value function both close to the optimum and for values that deviate significantly from the optimum. Therefore, we introduce a lower limit $R_{\min}$ for the reward, so that the reward is given by

$$R(y,u) = \max\{R_{\min}, -\text{cost}(y,u)\}.$$

Furthermore, each action consists of making some particular change to the parameter vector θ. The action set $\mathcal{A}$ is designed to contain nine different actions, eight of which correspond to a positive or negative change of each entry in θ, and the ninth one corresponding to not changing anything. The size of the change in each case is determined by a relative step size $\delta^n \in [0,1)$. The superscript n denotes the iteration number of the algorithm, and indicates that this step size can be changed between iterations. For the action a corresponding to a positive change of θ_i, we get

$$\theta_i \leftarrow \theta_i + \delta^n(\theta_i^{\max} - \theta_i^{\min}), \tag{4}$$

and for the action corresponding to a negative change of θ_i, we get the same but with the plus sign replaced by a minus sign.

The idea of the algorithm is to use an epsilon-greedy policy to select an action $a \in \mathcal{A}$ corresponding to a change in the parameter vector θ, run the system with the test inputs for this parameter value, and evaluate the result. The resulting reward is saved and then used to train the neural network that approximates the value function $V(\theta)$. This is then repeated with a decreasing epsilon, corresponding to decreasing the degree of exploration and increasing the degree of exploitation, so that more and more of the iterations lead to a high reward. The step size δ^n is also decreased with time. The purpose of this is that a large part of the state set $\mathcal{S}$ should be covered in the initial exploration phase, but that the changes should be small enough to find parameter values close to the optimum in the later exploitation phase.

The algorithm is listed in Algorithm 4. The first requirement for using the algorithm is to define the structure of the controller $K(\theta)$. This includes the two PI controllers $C_1(s)$ and $C_2(s)$, the parameter limits $\theta \in \mathcal{S}$, as well as the scaling matrices S^1 and S^2. We also have to specify the cost of measurement signals y and control signals u resulting from a particular run of the controller, consisting of applying test inputs as in Fig. 12 to all closed-loop inputs u_i^c. For this, we use the cost defined in the previous section. Also the lower limit for the reward, $R_{\min}$, must be specified. The amplitudes of the test input steps as well as their duration, given by $[t_i^0, t_i^f]$, must be provided as well. We also define an epsilon function ε^n and a step size function δ^n that are non-increasing functions of the iteration number n.

The experience gained from iteration n of the algorithm can be summarized as a pair (θ^n, R^n) of the new parameter value and the corresponding reward. This pair is stored in a replay buffer $\mathcal{B}$ to be used to train the value function neural network. The buffer keeps data from the B latest iterations. After each iteration, b samples are selected (with replacement) from this buffer and used to train the neural network. The purpose of this procedure is to make better use of the data from previous iterations, by reusing it in the training step, and to improve the convergence properties of the algorithm by using the previous samples in a random order and thus avoiding using unnecessarily correlated samples consecutively for updating the function approximating neural network. The reason for not keeping too old samples is that most of these correspond to a lower reward, and thus they are less relevant when the goal is to approximate a value function that should be used to maximize the reward. Discarding old data also allows the controller to adapt to process changes.

Further information that must be provided to the algorithm is (1) the learning rate η used when updating the weights in the neural network during training and (2) the structure of the network, e.g., the number of layers and the number of nodes in each layer.

As seen in Algorithm 4, the neural network is first initialized with some random weights ω^0. The initial parameter vector is $\theta = (1,0,0,1)^{\mathrm{T}}$, corresponding to a diagonal decoupling matrix D. In each iteration, a random action is chosen from the action set $\mathcal{A}$ with probability ε^n, and the action that leads to a new parameter vector θ^n that maximizes the current value function approximation $V(\theta;\omega^{n-1})$ is chosen with probability $1-\varepsilon^n$. The random choice is implemented by generating a random number r from a uniform probability distribution $U(0,1)$ on the interval $[0,1]$ and choosing the first case if $\varepsilon^n < r$ and the second case otherwise.

After choosing the action in iteration n, the parameter is updated according to the chosen action a, by the update function $\theta^{\text{update}}(\theta,\delta^n,a)$ that works as described in and around equation (4). Then, the system is run with the test inputs using the controller defined by θ^n. The reward is computed to evaluate the parameter value θ^n, and saved in the replay buffer $\mathcal{B}$. After this, b samples are randomly selected from the buffer (with replacement) and used to update the weights ω^n of the neural network according to the gradient descent method with a gradient step size given by the learning rate η, so that the value function approximation $V(\theta,\omega^n)$ better represents the values in the samples.

When enough exploration has been carried out, the value of ε^n as well as of the step size δ^n should be small, and the parameter values θ^n resulting from each iteration should remain close to the optimal value of θ.

5. Test Problem – Linearized Model

To test the method, we use a Modelica model of the system that is linearized around an operating point. The model has in total 162 states. At the stationary operating

Algorithm 4 Reinforcement Learning Algorithm

Input: Controller structure $K(\theta)$, cost function $\text{cost}(y,u)$, minimum reward $R_{\min}$, test inputs u^c, epsilon function ε^n, step size function δ^n, replay buffer size B, batch size b, learning rate η, and neural network settings

Output: Value function $V(\theta)$

Build network $V(\theta;\omega^0)$ with random weights ω^0
Initialize replay buffer $\mathcal{B}$
Initialize parameter vector $\theta^0 = (1,0,0,1)^{\mathrm{T}}$
for iteration $n = 1,2,\ldots$ **do**
    Get a random number $r \sim U(0,1)$
    if $r < \varepsilon^n$ **then**
        $a^n \sim \text{rand}(\mathcal{A})$
    else
        $a^n = \text{argmax}_{a\in\mathcal{A}} V(\theta^{\text{update}}(\theta^{n-1},\delta^n,a);\omega^{n-1})$
    end if
    Parameter update $\theta^n \leftarrow \theta^{\text{update}}(\theta^{n-1},\delta^n,a^n)$
    Run system with controller $u(t) = K(y(t);\theta^n)$ and test inputs u^c
    Get reward $R^n = \max\{R_{\min}, -\text{cost}(y,u)\}$
    Add (θ^n, R^n) to $\mathcal{B}$
    Sample tuples $\{(\theta^k,R^k)\}_{k=1}^b$ from $\mathcal{B}$
    Update weights $\omega^n \leftarrow \omega^{n-1} + \eta\sum_{k=1}^b \left[R^k - V(\theta^k;\omega^{n-1})\right]\nabla V(\theta^k;\omega^{n-1})$
end for

Table 4. Values of control variables as well as flows and temperatures of the evaporator and condenser water at the used linearization point. The listed variables are compressor frequency (compressor), expansion valve opening (exv), evaporator entering water temperature (eEWT), evaporator leaving water temperature (eLWT), evaporator water flow (eFlow), condenser entering water temperature (cEWT), condenser leaving water temperature (cLWT) and condenser water flow (cFlow).

compressor	**exv**
150 Hz	0.70

eEWT	**eLWT**	**eFlow**	**cEWT**	**cLWT**	**cFlow**
12°C	7°C	0.085 m^3/s	30°C	35°C	0.112 m^3/s

point, the control signals are kept at constant values. We use a point where the compressor frequency (u_2) is set to 150 Hz and the expansion valve opening (u_1) is 0.7, i.e., the valve is 70% open. For the stationary point corresponding to these inputs, the temperatures and flows of the water that circulates through the evaporator and compressor are listed in Table 4.

To evaluate the solution obtained from the examined method, which we refer to as *model-free decoupling control*, we compare it with two alternative approaches. The first one is decentralized PI control, where we use two separate single-input

Table 5. PI-controller parameters used for the different control methods.

control method	k_1^{P}	k_1^{I}	k_2^{P}	k_2^{I}
SISO	0.08854	0.01337	0	-12.8
model-based decoupling	2.622	0.6745	0	0.1708
model-free decoupling	$\frac{0.08854}{S_1^1 S_1^2}$	$\frac{0.01337}{S_1^1 S_1^2}$	0	$\frac{-12.8}{S_2^1 S_2^2}$

single-output (SISO) controllers, each using only one of the feedback signals to determine one of the control signals, as shown in Fig. 7. We refer to this case as *SISO control*. The second control approach is to use a decoupled controller, where the decoupling matrix is the inverse of the transfer function matrix that describes the input-output system $P_{\mathrm{io}}(s)$ with our two control signals u_1 and u_2 as inputs, and the feedback signals y_1 and y_2 as outputs, evaluated at $s = 0$. Thus, we use the controller structure shown in Fig. 8 and get a multiple-input multiple-output (MIMO) controller

$$\begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix} = P_{\mathrm{io}}^{-1}(0) \begin{bmatrix} C_1(s) & 0 \\ 0 & C_2(s) \end{bmatrix} \begin{bmatrix} E_1(s) \\ E_2(s) \end{bmatrix},$$

with the decoupling matrix $D = P_{\mathrm{io}}^{-1}(0)$. We refer to this case as *model-based decoupling control*. The model-based decoupling matrix is computed to

$$D_{\mathrm{model}} = P_{\mathrm{io}}^{-1}(0) = \begin{bmatrix} 0.01212 & -0.2153 \\ -3.222 & -53.86 \end{bmatrix}.$$

The PI controllers C_1 and C_2 used for each of the methods are selected using the MATLAB function `pidtune`. One set of PI controllers is selected for the SISO case, and another set of controllers for the model-based decoupling case. For the model-free decoupling case, the PI controllers are selected such that the closed-loop system is equal to the SISO control system when the scaled decoupling matrix $\widehat{D}$ (see (3)) is equal to the identity matrix, i.e., $\theta = (1,0,0,1)^{\mathrm{T}}$, which is the case when the algorithm is initialized. Parameterizing the PI controllers as

$$u_j(t) = k_j^{\mathrm{P}} e_j(t) + k_j^{\mathrm{I}} \int_0^t e_j(\tau)\, \mathrm{d}\tau, \quad j = 1,2,$$

where $e_j(t) = r_j(t) - y_j(t)$ is the control error, the used parameters k_j^{P} and k_j^{I} are listed in Table 5. Note that k_2^{I} is negative in the SISO case. This is explained by the fact that increasing the compressor speed (u_2) usually causes a decrease in eLWT (y_2), since the cooling effect increases. In the model-based decoupling case, this sign switch is incorporated into the decoupling matrix.

When running the model-free decoupling algorithm, we use the decoupling matrix factorization (3) with scaling matrices

$$S^1 = \begin{bmatrix} 1 & 0 \\ 0 & 150 \end{bmatrix} \quad \text{and} \quad S^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The choice of the first scaling matrix reflects that the typical range of the compressor speed is $[0,150]$, while the range of the expansion valve opening is $[0,1]$. Furthermore, we assume that the entries of the scaled decoupling matrix, i.e. our parameters to be determined, satisfy

$$\theta_i \in [-10,10], \quad \text{for } i = 1,2,3,4.$$

In the cost function, the only measurement signals that are considered are the feedback signals y_1 and y_2, i.e., the number of measurement signals is $m = 2$. Furthermore, we use the weight values $\alpha = 30$, $\beta = 1000$ and $\gamma = 150$. The tracking specification curves $y_{\min}$ and $y_{\max}$ are chosen as in Fig. 13 (scaled by the step height for each test input) for both feedback signals. This means that no cost is added if the step responses are faster than the step response of a first-order system with a time constant of 10 seconds and if there are no overshoots. The control limits for the expansion valve opening u_1 are chosen to $u_1 \in [-0.3,0.3]$, since the linearization is around the value 0.7, and the maximum valve opening in the nonlinear model corresponds to the value 1. The control limits for the compressor speed u_2 are set to $u_2 \in [-150,60]$, since the linearization is around the value 150, and the compressor speed must be non-negative, but should not be too high. The control weights are set to $w_1^u = 1/0.3$ and $w_2^u = 1/50$, to take into account the different ranges of the control signals. The output weights are chosen as $w_1 = w_2 = 1$, and the lower limit of the reward is selected as $R_{\min} = -15$.

For the test inputs, as shown in Fig. 12, we set the duration $t_i^f - t_i^0$ to 500 seconds for all i. The step amplitude is set to 1 for $i = 1,2,3$, corresponding to the subcooling reference r_1, the eLWT reference r_2 and the eEWT disturbance d_1. For $i = 4$, corresponding to the eFlow disturbance d_2, the amplitude is 0.01.

The functions ε^n and δ^n that should decrease with the iteration number n are set to decrease exponentially, so that

$$\varepsilon^n = \varepsilon_{\min} + (\varepsilon_{\max} - \varepsilon_{\min})e^{-n/T_\varepsilon} \quad \text{and} \quad \delta^n = \delta_{\min} + (\delta_{\max} - \delta_{\min})e^{-n/T_\delta}.$$

Their values start at $\varepsilon_{\max}$ and $\delta_{\max}$, and approach asymptotically $\varepsilon_{\min}$ and $\delta_{\min}$. The rate of decrease is determined by the time constants T_ε and T_δ. We use the values $\varepsilon_{\max} = 1$, $\varepsilon_{\min} = 0.01$, $T_\varepsilon = 2000$ for the ε^n function, and $\delta_{\max} = 0.1$, $\delta_{\min} = 0.01$, $T_\delta = 1000$ for the δ^n function.

The state and reward from each iteration are stored in a data buffer of size $B = 5000$. After each iteration, $b = 500$ values are randomly selected (with replacement) from this buffer and used to train the value function network $V(\theta;\omega)$. This network is selected to have 3 inner layers, each with 16 nodes. All layers have a ReLU activation function, except for the output layer that does not have an activation function. For training, we use the Adam optimizer with learning rate $\eta = 0.001$.

Table 6. Resulting reward for the different controllers.

control method	SISO	model-based decoupling	model-free decoupling
reward	−6.64	−4.61	−3.70

6. Results

The method was applied to the example setup described in the previous section. The reward obtained at each iteration is shown in Fig. 14. A smoothed version of the same data, using a 50-step moving average, is shown in Fig. 15. We see that after some initial exploration during the first 1000 iterations, the algorithm finds a solution that gives a relatively high reward. Some additional exploration is done for the following 500–1000 iterations, until the reward is stabilized around the highest found value for almost all remaining iterations.

The corresponding parameter values for each iteration are shown in Fig. 16. Here, the initial exploration where large changes are tried out during the first 1000 iterations is clearly visible. After 2000 iterations, the remaining variations are small, and the parameters remain close to values that correspond to the largest obtained reward.

The final reward, after 5000 iterations, is in Table 6 compared to the reward obtained with SISO control and with model-based decoupling control. We see that the method manages to find a decoupling matrix that results in better control performance than the alternative methods, in terms of the cost function used.

The different non-zero terms of the reward for the different control methods are shown in Fig. 17. Note that the scale on the y-axis is upside down, and that a lower bar corresponds to a higher reward and thus is better. We see that the model-based decoupling control gives better or similar performance compared to SISO control for all parts of the reward. The model-free decoupling control gives much better performance than the other methods for some of the terms, and performance similar to the model-based decoupling control for almost all of the remaining terms. Out of the three terms for which the model-free decoupling control performs noticeably worse than the model-based decoupling control, two consist in that the control signal derivative varies slightly more. This part of the reward is mainly a regularization to make sure that we avoid too rapid control signal variations, and these small reward differences should not lead to any significant disadvantage, as long as the actuators can be adjusted sufficiently fast. The third term in which the model-free decoupling control performs worse is the effect of the subcooling reference change on the eLWT. Although model-based decoupling is better in this case, the reward is rather large in both cases, compared to the other terms, and the performance is better than for the SISO controller even for this term. Overall, the method seems to be effective in finding a controller that gives small or at least similar values in all terms of our defined cost function, compared to the values obtained using the two reference controllers.

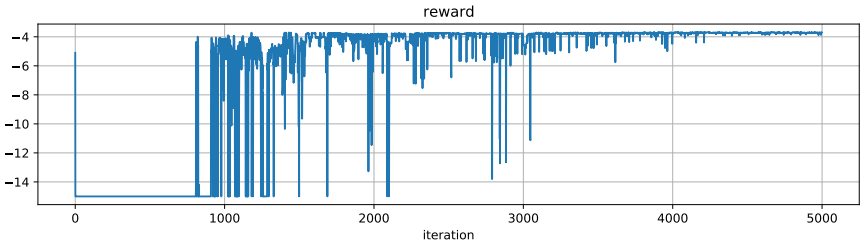


Figure 14. Reward at each iteration of the algorithm when starting out from a decentralized controller.

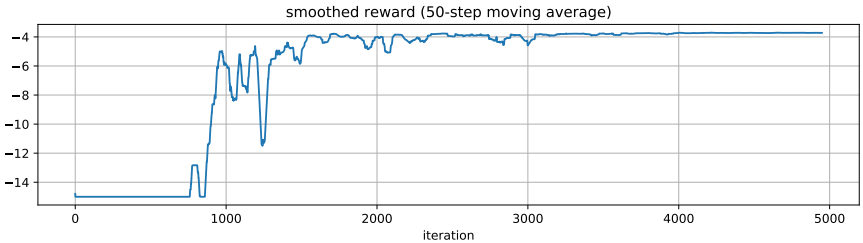


Figure 15. 50-step moving average of reward at each iteration of the algorithm when starting out from a decentralized controller.

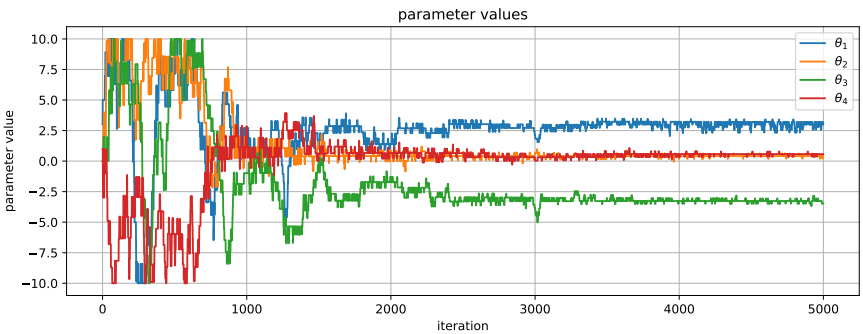


Figure 16. Parameter values θ defining the decoupling matrix at each iteration of the algorithm when starting out from a decentralized controller.

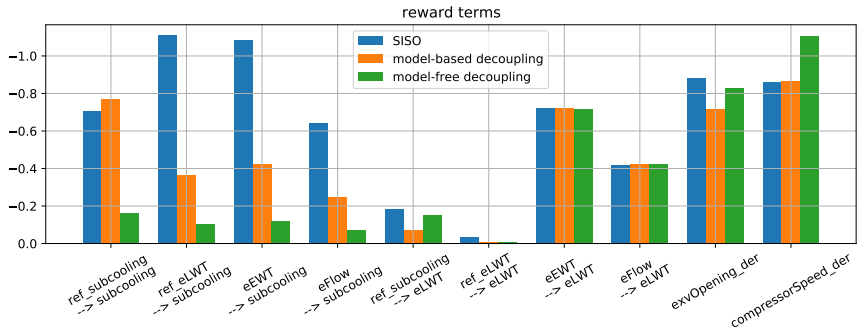


Figure 17. Values of the non-zero terms of the reward of the final solution. Values closer to zero are better. The first eight terms correspond to the effect on the reward of changes in the measurement signals subcooling and eLWT due to reference changes and disturbances. The last two terms correspond to the effect on the reward of control signal derivative variations.

The test inputs to the closed-loop system used are shown in Fig. 18. The resulting measurement and control signals are shown in Fig. 19. For the first measurement signal, the subcooling, both decoupling methods contribute significantly to reducing the deviations from the nominal value compared to the SISO case. The effect is clearly strongest when the model-free control is used. For the second measurement signal, eLWT, the overall performance is similar between all three methods, although the decoupling methods give a slightly higher reward.

In Fig. 20, we show the evaporator refrigerant level, eLevel. This signal is usually not measurable and is not used for control or in the design of the controller. However, we know that it is one of the signals for which it is important to keep the variations small, and it is therefore interesting to examine. We see that the effect on this signal from the model-free decoupling control is very similar to the effect from the SISO control for our test example. The performance for the model-based decoupling control is, however, slightly worse with respect to this signal.

Finally, we look at the control signals exvOpening and compressorSpeed in Fig. 19. We see that the signals satisfy the bounds that we defined in the cost, which means that the corresponding cost terms are zero. For the change of the subcooling reference, during the first 500 seconds, the model-free decoupling leads to control signals that are quite large during a short time, compared to the other methods. However, during the same time interval, the deviation in the exvOpening from the nominal value is smaller for the model-based decoupling control than for the SISO control, so the relatively large control signals do not seem to be a necessary consequence of a decoupled controller that performs better than the SISO controller. If one would like to decrease this effect, it should therefore be possible to do that by taking it into account in the cost function, while still maintaining relatively good

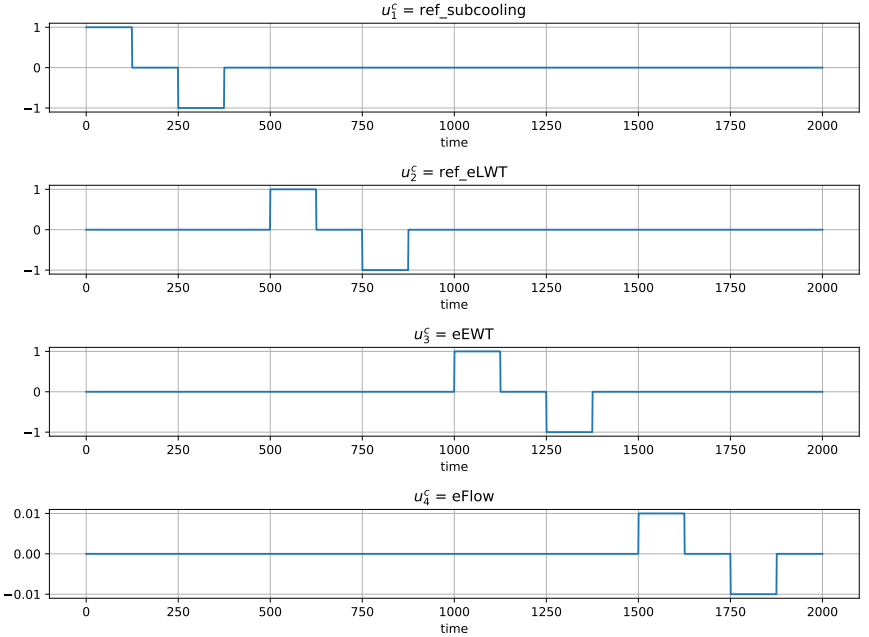


Figure 18. Test inputs for the linearized closed-loop system model. The shown inputs correspond to deviations from the operating point at which the model was linearized.

performance. For the rest of the simulation time, the decoupling controllers lead to control signals that are similar to the SISO control signals, but with smaller overshoots. The difference is clearest for the model-free decoupling control, where the control signals do not have any overshoot at all during this time interval.

The resulting scaled decoupling matrix for the model-free case is

$$\widehat{D}_{\mathrm{RL}} = \begin{bmatrix} 2.9880 & 0.4071 \\ -3.4801 & 0.5736 \end{bmatrix}.$$

Note that the scaled decoupling matrices for the different cases are not directly comparable since different PI controllers and scaling matrices are used. To compare the controllers, we look at the complete controller K, where $U(s) = K(s)E(s)$, for the Laplace transform of the control signal vector $U(s) = (U_1(s), U_2(s))^{\mathrm{T}}$ and the Laplace transform of the control error vector $E(s) = (E_1(s), E_2(s))^{\mathrm{T}}$. The complete controllers for the SISO case, K_{SISO}, the model-based decoupling case, K_{model}, and

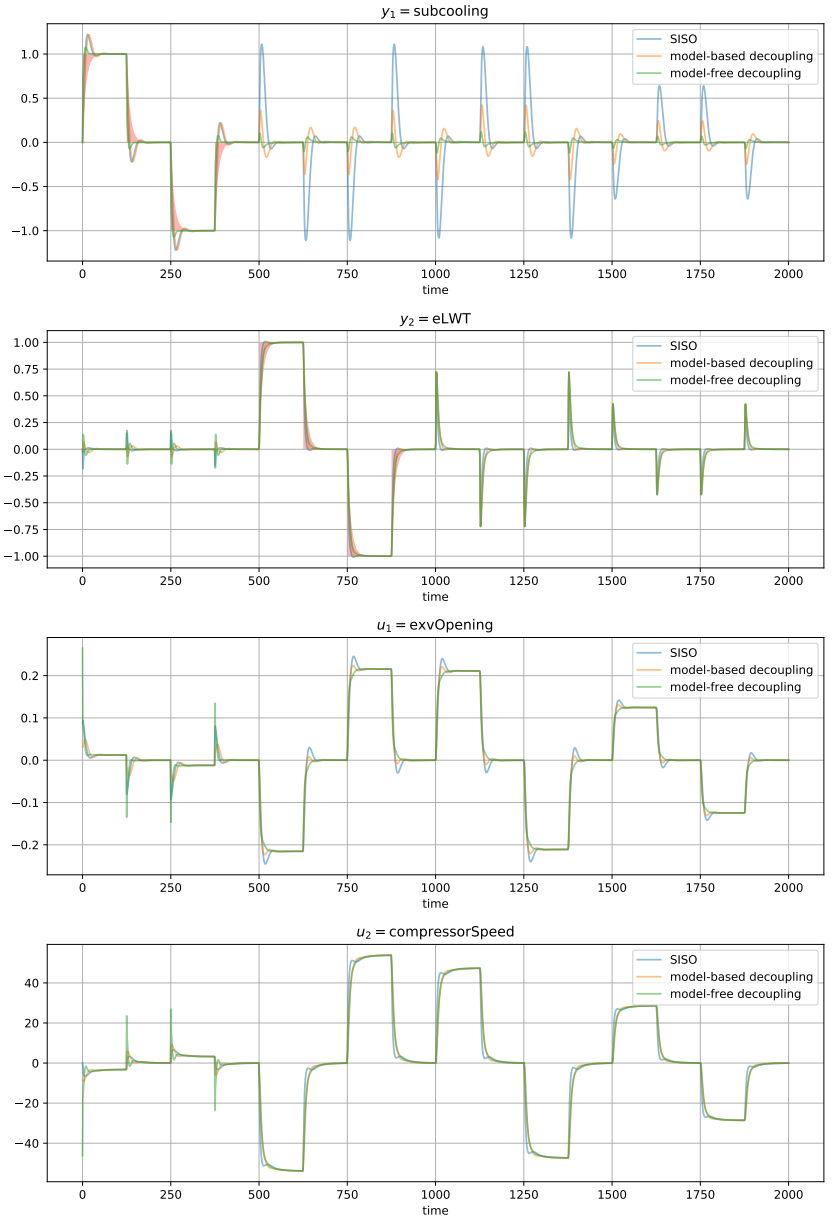


Figure 19. Measurement signals and control signals. The shown signals correspond to deviations from the operating point at which the model was linearized.

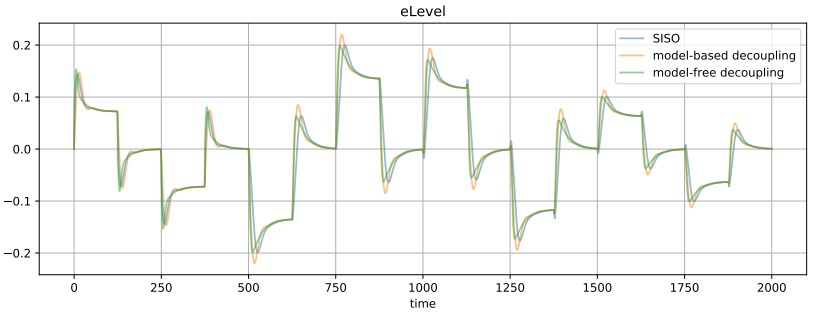


Figure 20. Evaporator refrigerant level. The shown signal corresponds to deviations from the operating point at which the model was linearized.

the model-free case, K_{RL}, are given by

$$K_{\text{SISO}} = \begin{bmatrix} 0.08854 + \frac{0.01337}{s} & 0 \\ 0 & -\frac{12.8}{s} \end{bmatrix},$$

$$K_{\text{model}} = \begin{bmatrix} 0.03178 + \frac{0.008175}{s} & -\left(8.448 + \frac{2.173}{s}\right) \\ -\frac{0.03677}{s} & -\frac{9.199}{s} \end{bmatrix},$$

$$K_{\text{RL}} = \begin{bmatrix} 0.2646 + \frac{0.03995}{s} & -\left(46.22 + \frac{6.979}{s}\right) \\ -\frac{0.03474}{s} & -\frac{7.342}{s} \end{bmatrix}.$$

We see that the decoupled controllers compute the second control signal, compressor speed, in similar ways, while the model-free decoupling controller has a clearly higher gain for computing the first control signal, expansion valve opening. The differences between the decoupled controllers and the SISO controller are very significant, with large diagonal elements for the decoupled controllers that clearly affect the control behavior significantly.

7. Discussion

This report presents a method for constructing a multiple-input multiple-output (MIMO) controller for a chiller process, with the property that it tracks reference changes while keeping variations in other relevant signals small. We would like the method to be able to automatically adapt the controller to new variants of the process. This should preferably be possible without needing an accurate model of the process to compute the controller.

Such a method was presented and tested on a simulated process, consisting of a model that was linearized around a specific operating point. The method uses a simple reinforcement learning approach in order to determine a decoupling matrix that in combination with two PI controllers makes up a MIMO controller with two

feedback signals and two control signals. The resulting controller was compared with a controller with the same structure, but where the decoupling matrix instead was computed using the static gain of an input-output model of the system, i.e., model-based decoupling. It was also compared to a decentralized controller, consisting of two separate single-input single-output (SISO) PI controllers, each using only one feedback signal to determine one control signal.

The performance of the controllers is measured in terms of a cost that is designed to represent the aim of keeping variations in measurement signals small while maintaining sufficiently good tracking and disturbance rejection performance. In terms of this cost, the MIMO controllers perform significantly better compared to the decentralized SISO controller structure. Furthermore, the model-free reinforcement learning based approach generates a controller that outperforms the model-based controller as well. This method also allows for the possibility to adapt the controller based on measurements after applying some test inputs. Thus, we have achieved the goal of creating a method for construction of a model-free adaptive MIMO controller that performs better than two separate SISO controllers.

Since the method was tested on a simulated system model, it was possible to run many iterations of the RL algorithm relatively quickly. With the settings in the test problem, we note that it takes around 2500 iterations before the parameter values and the reward have stabilized close to their final values. Running this many iterations on the real process would be very time consuming. The question is therefore how this method could be used practically.

First of all, we note that we have not put much effort into optimizing the performance of the RL algorithm. The purpose of this study has been to demonstrate that the method is a possible approach for determining a decoupling in a MIMO control setting. Therefore, we have chosen some reasonable default settings for the algorithm but have not emphasized trying many different combinations of these settings. Since there are many different settings to adjust, it is probable that selecting these with care could lead to significantly increased performance. Settings that could impact the performance are, e.g., the functions ε^n and δ^n that describe how the ε in the epsilon-greedy method is decreased, and how the step size for changing the parameters in each iteration is decreased. The amount of training data b used in each iteration, as well as the structure and settings of the neural network that represents the value function, are also factors where many variants could be tested and could impact performance. Future work would thus be to examine different combinations of these algorithm settings in order to optimize the performance.

Secondly, we must keep in mind that we initialize the method from a situation corresponding to the fully decentralized control case. The purpose of this is to demonstrate that even starting from a case with no decoupling structure at all, the algorithm is able to find a well-performing decoupling matrix. When using the algorithm in practice, it is more realistic to assume that we initialize the algorithm using parameters that are closer to optimal. When the algorithm is used to find controllers for many different but similar processes, we can initialize it with the values from

a previous similar process, instead of assuming no decoupling at all initially. If we have a model of the process that can be used for simulating it, we can also first run the algorithm on the model, to find parameter values that are close to optimal, and then run some iterations on the real process to optimize it further.

For practical use on a real process, one also has to take into account that the controllers resulting from the algorithm should be safe to run. One way to accomplish this could be to define the limits of the parameters θ so that any choice from the set of allowed parameter vectors leads to a safe controller. The question of how to determine such limits has not been considered in this report. However, even if this is not done, the real process should deactivate the controller and switch to another safe controller if some constraints are about to be exceeded, which, in any case, would lead to a low reward.

Besides examining different algorithm settings and the effect of different initialization points, future work should involve trying out the algorithm on a nonlinear model and eventually on real processes in the lab. Testing the algorithm on a nonlinear system, it is interesting to examine how the result differs depending on the operating points. An interesting question is whether it would be advantageous to use several different controllers for different operation ranges, i.e. to use gain scheduling, or if one controller would be sufficient for all operating points.

8. Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The work was also supported by the ELLIIT Strategic Research Area.

References

Afram, A. and F. Janabi-Sharifi (2014). "Theory and applications of hvac control systems–a review of model predictive control (MPC)". *Building and Environment* **72**, pp. 343–355.

AHRI (2017). *Air-Conditioning Heating and Refrigeration Institute. 210/240 performance rating of unitary air-conditioning and air-source heat pump equipment*.

Anderson, M., P. Young, D. Hittle, C. Anderson, J. Tu, and D. Hodgson (2002). "Mimo robust control for heating, ventilating and air conditioning (HVAC) systems". In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 1, 167–172 vol.1. DOI: 10.1109/CDC.2002.1184486.

ASHRAE (2021). *Guideline 36-2021 – High-performance sequences of operation for HVAC systems*.

Bellanco, I., E. Fuentes, M. Vallès, and J. Salom (2021). "A review of the fault behavior of heat pumps and measurements, detection and diagnosis methods including virtual sensors". *Journal of Building Engineering* **39**, p. 102254.

Breuker, M. S. and J. E. Braun (1998). "Common faults and their impacts for rooftop air conditioners". *HVAC&R Research* **4**:3, pp. 303–318.

Burns, D. J., C. R. Laughman, and M. Guay (2018). "Proportional-integral extremum seeking for vapor compression systems". *IEEE Transactions on Control Systems Technology*. ISSN: . DOI: 10.1109/TCST.2018.2882772. URL: https://www.merl.com/publications/TR2019-032.

Carrier (n.d.). *Carrier Product Data AquaEdge High-Efficiency Semi-Hermetic Centrifugal Liquid Chillers*. https://www.carrier.com/commercial/en/us/products/chillers-components/water-cooled-chillers/19mv/\#documents. [Online; August, 2022].

CEN (2018). *European Committee for Standardization. EN 14825, Air conditioners, liquid chilling packages and heat pumps, with electrically driven compressors, for space heating and cooling - testing and rating at part load conditions and calculation of seasonal performance*.

Chen, W., C. Zhi-jiu, Z. Ruiqi, and W. Yezheng (2002). "Experimental investigation of a minimum stable superheat control system of an evaporator." *International Journal of Refrigeration* **25**, pp. 1137–1142.

Chen, Y., S. Deng, X. Xu, and M. Chan (2008). "A study on the operational stability of a refrigeration system having a variable speed compressor". *International Journal of Refrigeration* **31**:8, pp. 1368–1374. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2008.04.012. URL: https://www.sciencedirect.com/science/article/pii/S0140700708000753.

Cheung, H. and J. E. Braun (2013). "Simulation of fault impacts for vapor compression systems by inverse modeling. part ii: system modeling and validation". *HVAC&R Research* **19**:7, pp. 907–921.

Choi, J. and Y. C. Kim (2002). "The effects of improper refrigerant charge on the performance of a heat pump with an electronic expansion valve and capillary tube". *Energy* **27**:4, pp. 391–404.

Dong, L., Y. Li, B. Mu, and Y. Xiao (2015). "Self-optimizing control of air-source heat pump with multivariable extremum seeking". *Applied Thermal Engineering* **84**, pp. 180–195. ISSN: 1359-4311. DOI: https://doi.org/10.1016/j.applthermaleng.2015.03.038. URL: https://www.sciencedirect.com/science/article/pii/S1359431115002616.

Dong, L., Y. Li, T. I. Salsbury, J. M. House, and Z. Wu (2018). "Multi-variable extremum seeking control for a multi-functional variable refrigerant flow system". *Science and Technology for the Built Environment* **24**:4, pp. 382–395. DOI: 10.1080/23744731.2017.1393257.

Elliott, M. S. and B. P. Rasmussen (2010). "On reducing evaporator superheat nonlinearity with control architecture". *International Journal of Refrigeration* **33**:3, pp. 607–614. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2009.12.013. URL: https://www.sciencedirect.com/science/article/pii/S0140700709002904.

Emerson (n.d.). *Copeland Compressor Product Data - ZPV0662E*. https://webapps.emerson.com/online-product-information. [Online; August, 2022].

Gordon, B. W. and H. Asada (2000). "Modeling, realization, and simulation of thermo-fluid systems using singularly perturbed sliding manifolds". *J. Dyn. Sys., Meas., Control* **122**:4, pp. 699–707.

Goswami, D., G. Ek, M. Leung, C. Jotshi, S. Sherif, and F. Colacino (2001). "Effect of refrigerant charge on the performance of air conditioning systems". *International journal of energy research* **25**:8, pp. 741–750.

Goyal, A., M. A. Staedter, and S. Garimella (2019). "A review of control methodologies for vapor compression and absorption heat pumps". *International Journal of Refrigeration* **97**, pp. 1–20.

Hägglund, T. (2012). "Signal filtering in pid control". *IFAC Proceedings Volumes* **45**:3, pp. 1–10.

He, X.-D., S. Liu, and H. H. Asada (1997). "Modeling of Vapor Compression Cycles for Multivariable Feedback Control of HVAC Systems". *Journal of Dynamic Systems, Measurement, and Control* **119**:2, pp. 183–191. ISSN: 0022-0434. DOI: 10.1115/1.2801231. eprint: https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/119/2/183/5778899/183_1.pdf. URL: https://doi.org/10.1115/1.2801231.

Jackson, S. D., A. N. Palazotto, M. Pachter, and N. Niedbalski (2019). "Control of vapor compression cycles under transient thermal loads". In: *AIAA Scitech 2019 Forum*, p. 0536.

Jolly, P. G., C. P. Tso, P. K. Chia, and Y.-W. Wong (2000). "Intelligent control to reduce superheat hunting and optimize evaporator performance in container refrigeration". *HVAC&R Research* **6**, pp. 243–255.

Kim, M., W. V. Payne, P. A. Domanski, C. Hermes, et al. (2006). "Performance of a residential heat pump operating in the cooling mode with single faults imposed (nistir 7350)". *NIST*.

Koeln, J. P. and A. G. Alleyne (2014). "Optimal subcooling in vapor compression systems via extremum seeking control: theory and experiments". *International Journal of Refrigeration* **43**, pp. 14–25. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2014.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0140700714000632.

Li, B. and A. G. Alleyne (2010). "A dynamic model of a vapor compression cycle with shut-down and start-up operations". *International Journal of Refrigeration* **33**:3, pp. 538–552.

Li, B., N. Jain, and A. G. Alleyne (2012). "Lmi control design for nonlinear vapor compression cycle systems". In: *Dynamic Systems and Control Conference*. Vol. 45301. American Society of Mechanical Engineers, pp. 711–718.

Li, H. and J. E. Braun (2007). "Decoupling features and virtual sensors for diagnosis of faults in vapor compression air conditioners". *International Journal of Refrigeration* **30**:3, pp. 546–564.

Lu, H., X. Cui, H. Han, Y. Fan, and Y. Zhang (2022). "A feature importance ranking based fault diagnosis method for variable-speed screw chiller". *Science and Technology for the Built Environment* **28**:2, pp. 137–151.

Madani, H. and E. Roccatello (2014). "A comprehensive study on the important faults in heat pump system during the warranty period". *International Journal of Refrigeration* **48**, pp. 19–25.

Mitchell, J. and J. Braun (2012). *Principles of Heating, Ventilation, and Air Conditioning in Buildings*. Wiley. ISBN: 9780470624579. URL: `https://books.google.com/books?id=nXekZwEACAAJ`.

Morud, J. and S. Skogestad (1996). "Dynamic behaviour of integrated plants". *Journal of Process Control* **6**:2-3, pp. 145–156.

O'Hegarty, R., O. Kinnane, D. Lennon, and S. Colclough (2022). "Air-to-water heat pumps: review and analysis of the performance gap between in-use and product rated performance". *Renewable and Sustainable Energy Reviews* **155**, p. 111887.

Pršić, D. and A. Vičovac (2017). "Modelling of control logic for start-up and shut-down sequences of the heat pump". In: *7th International Conference on Information Society and Technology ICIST*.

Rasmussen, B. P. and B. Shenoy (2012). "Dynamic modeling for vapor compression systems—part ii: simulation tutorial". *HVAC&R Research* **18**:5, pp. 956–973.

Rasmussen, H., C. Thybo, and L. F. S. Larsen (2006). "Nonlinear superheat and evaporation temperature control of a refrigeration plant". *IFAC Proceedings Volumes* **39**, pp. 251–254.

Rehrl, J., M. Horn, and M. Reichhartinger (2009). "Elimination of limit cycles in hvac systems using the describing function method". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, pp. 133–139.

Schurt, L. C., C. J. Hermes, and A. T. Neto (2010). "Assessment of the controlling envelope of a model-based multivariable controller for vapor compression refrigeration systems". *Applied Thermal Engineering* **30**, pp. 1538–1546.

Shah, R., B. P. Rasmussen, and A. G. Alleyne (2004). "Application of a multivariable adaptive control strategy to automotive air conditioning systems". *International Journal of Adaptive Control and Signal Processing* **18**:2, pp. 199–221.

Shang, Y., A. Wu, and X. Fang (2015). "A study on the modeling of the minimal stable superheat for a variable speed refrigeration system". *International Journal of Refrigeration* **59**, pp. 182–189.

Skogestad, S. and I. Postlethwaite (2005). *Multivariable feedback control: analysis and design*. john Wiley & sons.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction, 2nd Ed.* MIT Press.

Tassou, S. and I. Grace (2005). "Fault diagnosis and refrigerant leak detection in vapour compression refrigeration systems". *International journal of refrigeration* **28**:5, pp. 680–688.

Uhlmann, M. and S. S. Bertsch (2012). "Theoretical and experimental investigation of startup and shutdown behavior of residential heat pumps". *International Journal of Refrigeration* **35**:8, pp. 2138–2149.

Vinther, K., H. Rasmussen, R. Izadi-Zamanabadi, and J. Stoustrup (2013). "Single temperature sensor superheat control using a novel maximum slope-seeking method". *International Journal of Refrigeration* **36**, pp. 1118–1129.

Xiao, J., W. Wang, Y. Zhao, and F. Zhang (2009). "An analysis of the feasibility and characteristics of photoelectric technique applied in defrost-control". *International Journal of Refrigeration* **32**:6, pp. 1350–1357.

Yang, M.-H. and R.-H. Yeh (2015). "Theoretical analysis of optimal subcooling for single vapor-compression refrigeration systems". *Heat Transfer Engineering* **36**:10, pp. 912–925.

Yun, Y. and Y. S. Chang (2021). "Refrigerant charge prediction of vapor compression air conditioner based on start-up characteristics". *Applied Sciences* **11**:4, p. 1780.

Zhang, Q., S. Stockar, and M. Canova (2015). "Energy-optimal control of an automotive air conditioning system for ancillary load reduction". *IEEE Transactions on Control Systems Technology* **24**:1, pp. 67–80.

Zheng, X., R. Shi, S. You, Y. Han, and K. Shi (2019). "Experimental study of defrosting control method based on image processing technology for air source heat pumps". *Sustainable Cities and Society* **51**, p. 101667.