UNIVERSITÀ DEGLI STUDI DI UDINE
hic sunt futura

Università degli Studi di Udine

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Dottorato di Ricerca in Informatica e Scienze Matematiche e Fisiche

Ph.D. Thesis

Optimization and Modeling Techniques for Food Service Appliances

Candidate:
Eleonora Pippia

Supervisor:
Prof. Alberto Policriti

Co-Supervisor:
Dr. Eng. Emidio Tiberi

Cycle XXXIII – Year 2021

Author's e-mail:
pippia.eleonora@spes.uniud.it
pippia.eleonora@gmail.com

Institute Contacts:
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine – Italia
https://www.dmif.uniud.it

Well, here at last, dear friends, on the shores of the Sea comes the end of our fellowship in Middle-earth. Go in peace! I will not say: do not weep; for not all tears are an evil.

— J. R. R. Tolkien, The Lord of the Rings

Abstract

In the last couple of years food service sector is embracing the fourth industrial revolution. Electrolux Professional, as a leader in this sector, is continuously searching for smart functionalities and intelligent products to improve the design process or directly the final products.

This thesis, funded by Electrolux Professional, starts the analysis in food service sector of the emerging type of systems called Cyber Physical Systems, i.e. systems where computational and dynamical/physical capabilities as deeply intertwined. We present four different analyses focusing on optimization problems and reachability. These two aspects even overlap in one case study.

We start proposing a new functionality for professional ovens able to sort a list of cooking recipes in order to minimize the total energy consumption. Then a multi-objective optimization problem to select the complex parameters for a thawing system. For this second project we propose two approaches: a multi-objective analysis using a genetic algorithm and a reformulation of the optimization problem as a reachability analysis over a hybrid system. With this second study we build a bridge between the optimization field and the tools used for verification analysis. We open here the second topic of this thesis and the next two projects deal with reachability analysis. We select a specific reachability tool based on the Bernstein approximation for polynomial functions. We apply the Bernstein theory to the trajectories of a robot arm in order to check the collision with an obstacle or to prevent the collision applying a set of constraints in the joint domain. Finally, the last analysis touches another important trend of these years: Neural Networks (NN). We focus on the verification of neural net control systems (NNCS), i.e. systems where the NN are used to control the physical process. The challenge here is to be able to estimate and bound the behavior of the neural net. We propose a method, to this end, using rational and polynomial approximations for activation functions in order to bound the output image of each layer using the Bernstein expansion.

Acknowledgments

It is hard to find the right words to thank all the people that helped me during this long journey of professional and personal growth, but it is also impressive to look back and concretely see the results of a beautiful team-working.

I want to thank first my supervisor, professor Alberto Policriti for the patient and the great positivity shown for each new project, even the most exotic ones, and to be able to believe in these works and see with foresight the path that we were taking. My gratitude goes also to Riccardo Furlanetto and Michele Simonato that gave me the opportunity to start this journey with Electrolux Professional and introduced me to the industrial world. Thanks to both to have inspired and guided me and to have strongly believed in the young, curios and passionate mathematician that I was 3 years and half ago (and I hope I'm still now). I also want to thank from the deep of my heart my co-supervisor and close friend Emidio Tiberi. Thanks for all his suggestions, for all the discussions at the whiteboard and for all the lessons on thermodynamics at 6pm. I can safely say that this milestone of mine would not have been possible without his support and encouragement.

A special thank goes to Thao Dang for her expertise, which helped me to substantially improve the manuscript adding an entire new topic. I want also to thank her for the very warming hospitality during my visit at the Université Grenoble Alpes in France: she helped me in the tough moment of organizing my way back during the global pandemic. Thanks also to the Verimag team and especially to Akshay Mambakam and Nikolaos Kekatos to have welcomed me and made me feel at home, even in such a short time.

Finally, I want to thank all those people who shared this path with me. It is a long list that I will try to summarize as much as possible. I begin with Bermet Kerim Kyzy that started this journey in advance with me, I want to thank her for teaching me a lot of the Kazakhstan culture and for pushing me to speak English from the beginning. Then I thank my office-colleague Manlio Valenti for all the stimulating conversations and break time playing together. Thanks also to Arianna Bozzato for the tea times and for sharing with me a part of the research project. Thanks to Yuri de Pra for the moral support and to have let me play with his "cat"-experiments and thanks to Débora Pereira for all the passion and enthusiasm that she has brought in any activity we faced together. Thanks also to the entire The Research Hub in Electrolux Professional, Mauro Sonego, the mechatronic team, Daniele Turrin, the colleagues in Ljungby and all the others that have crossed my road. And of course, I would like to thank my family and my beloved Andrea, without whose support and love none of this would have been possible.

Contents

List of Figures

List of Tables

Introduction

We are in the era of connectivity, the era of autonomous driving car and collaborative robot, the era of cloud services and powerful integrated circuit. We are in the forth industry revolution where factories are smart, products are intelligent, and customers are demanding for being all around served with great satisfaction. This revolution is changing people's life styles and living behaviors, even thinking and mindset, from shopping to dining, from working to entertaining.

In this context a new generation of hybrid systems called *Cyber Physical Systems* (CPS) [11] are becoming increasingly pervasive. The term CPS refers to physical and software components that are deeply intertwined. These systems are able to collect and analyze data generated from physical sensors and to use these information to act and modify the physical world around them in order to exhibit multiple and distinct behaviors/modalities and to interact with each other by spreading the information to other devices. The double nature of CPS also changes the way in which we design the controls for these systems. We build computational models in place of real prototypes which can be simulated in order to verify their compliance with respect to original requirements. Indeed, the simulation-driven design processes, so that the *Model-Based Design* (MBD), quickly assumed a central role in system design. Virtual prototyping is a key concept in the development phase and it has two outcomes. On the one side, the ability to simulate the system speeds up the testing phase, reducing time and cost, and it enables the use of more sophisticated optimization tools to analyze any possible scenario. On the other hand, the complexity of generated models poses new challenges on how to design safety and secure CPS and how to verify the requirements on the system. At the foundation, to address the two outcomes we have optimization theory and reachability analysis. These two aspects characterize the thesis and they even overlap in one of the case studies.

Food service sector

The fourth industrial revolution, that started in the automotive sectors, is now involving any type of business. In the last couples of years, food service sector is embracing this revolution too. Shifting consumer preferences from conventional to technologically advanced products is enabling manufacturers to offer innovative kitchen appliances. We can think about all the videos and spot showing the first attempt of collaborative kitchen robots, or smart appliances or artificial intelligence algorithms all around the kitchen to guide and assist the food operators.

Electrolux Professional is a multinational company in food service equipment and laundry solutions. It fulfills the business-to-business[1](B2B) HORECA[2] market by providing a comprehensive range of solutions to store, prepare, cook, serve food, and to clean the tools employed in the food service. Electrolux food service equipment is tailored for different types of professional kitchens from Michelin starred restaurants,

to pubs and bars, from hospitals, schools and military canteens to Quick Service Restaurants. As a leader in food-service equipment, Electrolux Professional is continuously innovating the products with advanced solutions to improve efficiency and productivity of professional kitchens.

This thesis starts the analysis of CPS in food service sector focusing on optimization problems and reachability analysis. We present four different analyses. In the first two, we improve and develop new solutions using optimization techniques with a direct impact on the products and on the development phase. In the last two analyses, we focus on the verification of advanced tools that are increasingly presents in professional kitchens, i.e. collaborative robots and neural network, and we select a specific reachability tool based on Bernstein theory for polynomial functions.

Thesis contribution & structure

As mentioned before, we work on two directions: optimization of CPS and reachability analysis. This thesis has been divided into three parts. The first part presents the basic concepts. The second and third parts refer to applications of optimization and reachability theories on some practical projects.

Part I (Basic Concepts). This part recalls all the necessary definitions and properties. In Chapter 1 we define what is an optimization problem and we describe a generic multistage system, that is a particular system present in food service appliances. We also go beyond the range of the standard optimization tools entering in the field of the artificial intelligence tools by introducing the neural network structure. The hybrid system theory and the Bernstein theory that we use for the reachability analysis are reported in Chapter 2.

Part II (Optimization of multistage systems). This part is devoted to the development and application of optimization algorithms for food service appliances. In particular we address two projects involving a professional oven and a thawing appliance. Professional appliances allow to program a working cycle by setting up one or more modes, instead of setting different heating stages manually. These types of cycles are called *multistage systems*.

In Chapter 3 we develop a new functionality for professional ovens, able to sort a list of multistage systems in order to minimize the total energy consumption. This optimization algorithm is patented [SFPT19] and it is currently implemented on Electrolux Professional ovens. The scheduling problem is reformulated as the standard optimization problem to find an Hamiltonian circuit on a graph (see in [LPR20] a parallel work for sufficient conditions for Hamiltonian, 4-regular graphs).

The project followed in Chapter 4 is a multi-objective optimization analysis for a single thawing multistage system. The availability of a virtual model that simulates the thawing appliance, allows to implement a multi-objective genetic algorithm

[1]Business-to-business marketing involves the commercial transaction of a company's product or service to another company.

[2]From international Union of National Associations of Hotel, Restaurant, and Cafe Keepers, HORECA is an abbreviation used in Europe to designate the food service Industry Market (Restaurants, Hotels, Bars And Cafes, Supermarkets, Hospitals And Care Homes, Business, Transport & Industry, Commercial Laundries, Self-Services Laundries).

[PBT20]. The obtained optimal multistage cycles have been tested by a Food Technology researcher to evaluate the results in terms of food properties of thawed chicken fingers [BPTM21]. Finally this project gave us the possibility to show the link between our two lines of research. Indeed in Chapter 4.2 we were able to reformulate the optimization analysis as a reachability problem thanks to the nature of the multistage systems [PBT$^+$19].

Part III (Reachability analysis of hybrid systems). In this last part we look ahead at the emerging trends on the kitchens, like collaborative robots and artificial intelligent algorithms, and we address the fundamental topics of verification and safety.

Essential for the verification of a control system is the reachability problem. We decided to select a specific reachability technique that uses the Bernstein coefficients to bound a polynomial or a rational function.

In Chapter 5 we start using this technique in the robotic field, where we address the problem to literally reach or avoid-to-reach obstacles with a robot arm.

Finally we touch the last emerging topic that is the use of neural networks as intelligent control systems. Indeed, Electrolux Professional is currently evaluating the potential of neural network controllers for food service appliances. In Chapter 6 we address the emerging need to verify and estimate the behavior of a neural networks. We propose a method to bound the output value of a neural network by approximating each layer with a rational or polynomial function and using the Bernstein theory [PDP20].

The structural choices have resulted in the following contents of the thesis, as visualized in Figure 1.

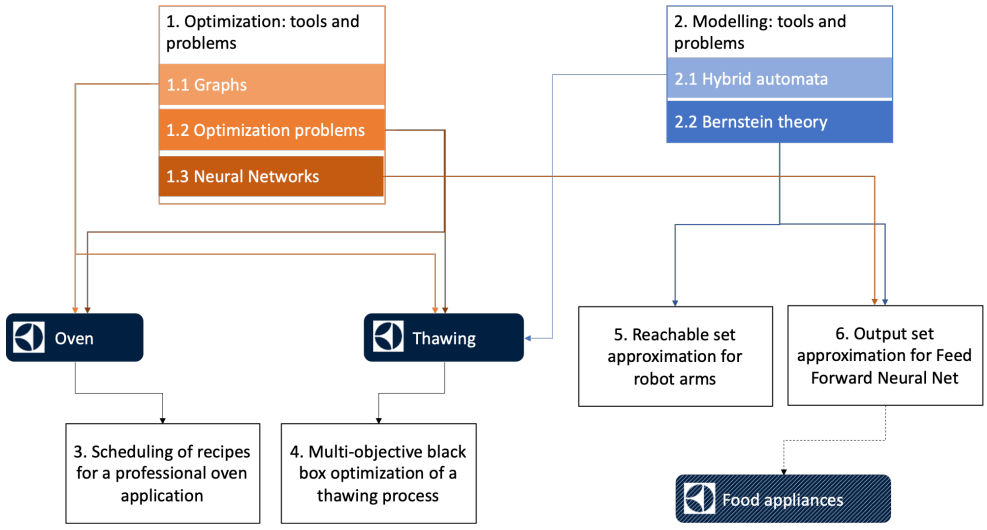


Figure 1: Schematic overview of the contents.

I

Basic Concepts

1

Optimization: tools and problems

Optimization deals with problems of minimizing or maximizing a function of several variables usually subject to constraints. In other words we are searching for the best combination of variables to achieve a certain goal. The applications of optimization are limitless both for industrial and academical topics. In industrial application, it is quite common that the range of variables are discrete or the constraints are relational constrains. In these cases the underline structure of the problem can be describe as a graph structure.

The system we want to analyze is a multistage cooking program and (qualitatively/ quantitatively) governing this kind of processes is fundamental for most professional food appliances. Professional chefs, in fact, use different heating methods to bring more flavor to the food or to control the final results. For example, in an oven they may start with a short grill for searing a steak and then continue the process at low temperature for a long time [14]. Or they could select in a fridge 3 different temperatures to cool down gently a cooked product defining a duration for each temperature.

Lifting up our view from the single application, a *multistage systems* is a system characterized by n stages, where each stage has a precise—albeit parametric—evolution law. The underlying structure is a *Directed Acyclic Graphs* (DAG). Lets first of all introduce definitions and properties both for graph theory and optimization theory that will be used throughout Part II, where we will optimize a multistage system and a list of multistage systems.

1.1 Graphs

Graph theory is not only an area of interest in its own right but it gives also a basic framework to describe the underline discrete structure of complex systems and models such as multistage systems. We introduce some notation and we leave [20] for a detailed introduction.

Definition 1.1.1. *A* graph *(or undirected graph) is a pair $G = (V, E)$, where V is a*

finite set of vertices or nodes and $E \subseteq V \times V$ is a set of edges.

Given and edge $e = \{v, v'\}$ we say that the edge e is incident in v and v'. We define $\delta(v)$ the set of edges incident on the node v and we call the *degree* of v the cardinality $|\delta(v)|$.

Definition 1.1.2. *(Regular graph) A graph is k-regular if every vertex has degree k.*

A particular class of graphs are the complete graphs K_n, graphs with n vertices and edges $\{i, j\}$ for each pair $i, j \in V$.

Definition 1.1.3. *(Path and cycle) Let $G = (V, E)$ be a graph. A* path *is a sequence of distinct vertices $(v_i)_{i\in[0,n]}$ and distinct edges $(e_i)_{i\in[0,n-1]}$ such that, $e_i = \{v_i, v_{i+1}\} \in E$ for all $i \in [0, n-1]$. A* cycle *is a path with the additional edge $\{v_0, v_n\} \in E$.*

A particular path that we can define on a graph is the *Hamiltonian path* that is a path that visits all the vertices of the graph. Similarly an *Hamiltonian cycle* is a cycle that visits all the vertices of the graph. Determine if such cycle exists in a graph is well known to be NP-complete (see [20] also for computational complexity).

Definition 1.1.4. *A graph is* direct *if E is a set of couples $e = (v, v')$, where v is the source and v' is the destination of the edge.*

We can easily extend the above definitions for direct graphs.

1.2 Optimization problems

We can now define what we mean with an *optimization problem.* A standard form of an optimization problem is

$$\begin{aligned} \mathscr{P}: \quad & \min_x \quad f(x) \\ & s.t. \;\; g_i(x) \leq 0 \quad i = 1, \ldots, k \end{aligned} \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^m$ are m *objective functions* and $g_i \leq 0$ are k constraints.

One interesting example of optimization problem over a graph structure is the following,

Definition 1.2.1. *(Traveling salesmen problem (TSP)) Given a graph $G = (V, E)$ and a weight function $w : E \to \mathbb{R}$, find an Hamiltonian cycle H such as $\sum_{e\in H} w(e)$ is minimum.*

This problem is extremely famous and can be found in many applications. If we consider a direct graph we can define the *Asymmetric TSP* where the weight to travel from one vertex to another is different if we travel on the opposite direction.

Another important class of optimization problems is the class of *job-shop* or *scheduling* problems in which multiple jobs are processed on one or several machines. Each job consists of a sequence of tasks, which must be performed in a given order.

For example, the job could be a multistage system and each stage is a specific task of the job. The problem is to schedule the tasks on the machines so as to minimize the length of the schedule (time) or total weight of the schedule (e.g. energy). It is interesting to notice that the TSP can be reformulated as a job shop problem, where each node is a job and we want to schedule all the jobs one after the other. Viceversa some job shop problems can be reformulated and solved as a TSP instance.

1.2.1 Multi-objective optimization problems

Especially in industry field, many decision and planning problems involve multiple conflicting objectives that should be considered simultaneously. For example, we want to find the best multistage system that maximize the final food result while minimizing the energy consumption of the appliance or the running water. Such problems are generally known as *multi-objective* optimization problem where the problem $\mathscr{P}$ in Equation (1.1) has $m > 1$.

Although single-objective optimization problems may have a unique optimal solution, multi-objective optimization problems (as a rule) present a possibly uncountable set of solutions. A key concept in determining a set of multi-objective optimization problem solutions is that of Pareto front.

Definition 1.2.2. *(Pareto Dominance) A vector* $\mathbf{u} = (u_1, \ldots, u_k)$ *is said to* dominate $\mathbf{v} = (v_1, \ldots, v_k)$ *(denoted by* $\mathbf{u} \preceq \mathbf{v}$*) if and only if* $\mathbf{u}$ *is partially less than* $\mathbf{v}$*, i.e.,* $\forall i \in \{1, \ldots, k\}\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\}\ u_i < v_i$.

Definition 1.2.3. *(Pareto Front) For a given multi-objective optimization problem with objective functions* $\mathbf{F}(\mathbf{x})$*, the* Pareto front P *is defined as:*

$$P := \{\mathbf{x} \mid \neg\exists \mathbf{x}' : \mathbf{F}(\mathbf{x}') \preceq \mathbf{F}(\mathbf{x})\}. \tag{1.2}$$

In order to evaluate a set of solutions, we need to use a performance indicator able to compare two different Pareto fronts. To conclude the introduction on multi-objective optimization problem we propose the *hypervolume indicator* I_H [66] as one possibility to evaluate the Pareto front. This indicator, that we will use in Chapter 4.1, measures the hypervolume of the portion of the objective space that is dominated by the set of points on the Pareto front. In order to measure this quantity, the objective space must be bounded or, alternatively, we should use a reference point that is at least weakly dominated by all points.

The hypervolume indicator in 2D is an area as in Figure 1.1. It is easy to be calculated dividing the area in slices (horizontally or vertically) and this partition can be calculated in linear time, processing the points on the Pareto front in ascending order of the first coordinate (Figure 1.2). Compute the hypervolume indicator in 2D is $\Theta(n \log n)$ using the slice partitioning because it is just necessary to reorder the first coordinate. [117] shows that also in 3D we can extend the slice partitioning idea to reach the same complexity as in Figure 1.3 where the bounding point is (r_1, r_2, r_3).

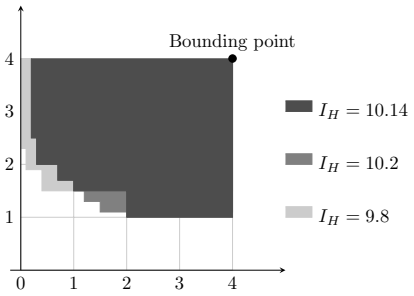


Figure 1.1: Three different hypervolumes (areas) in 2D

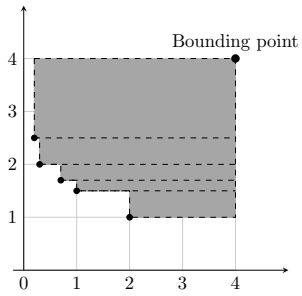


Figure 1.2: Horizontally slice division of a hypervolume (area) in 2D

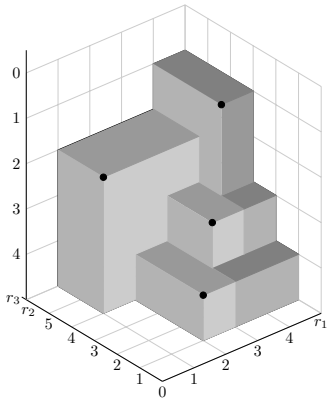


Figure 1.3: Slice division of a hypervolume (volume) in 3D

1.3 Neural networks

As we noted, graphs and networks can be used to model many important problems in engineering, business and the physical and social science. Machine learning algorithms are general-purpose tools for approximating models from data [18]. They are able to solve problems from many disciplines without detailed domain specific knowledge. Optimization lies at the heart of machine learning. The essence of most machine learning algorithms is to build an optimization model and learn the parameters in the objective function from the given data. The main steps of machine learning are the selection of the model, usually from an appropriate family of models, and the learning phase where we determine the parameters by minimizing an objective function (e.g. a loss function) of the training samples.

Artificial neural network (NN) is a popular machine learning family of models that simulates the mechanism of learning in biological organisms [1]. NNs have been around since 1943, when they were first introduced in Warren McCulloch and Walter Pitts' paper "A Logical Calculus of the Ideas Immanent in Nervous Activity", describing how networks of artificial neurons could be used to solve various logic problems. However, after the initial euphoria, there was a period of disappointment in which the data hungry and computationally intensive nature of neural networks were seen as an impediment of their usability.

Nowadays, in the era of connectivity, where we have more data than ever, and faster computer-processing, we can properly apply NNs in more practical settings. NNs plays a significant role in many fields, such as machine translation, speech recognition, image recognition, recommendation system, and they are increasingly used also in the development of control systems in many hybrid system applications such as robots, self-driving vehicles (see for example [119, 49] and references therein).

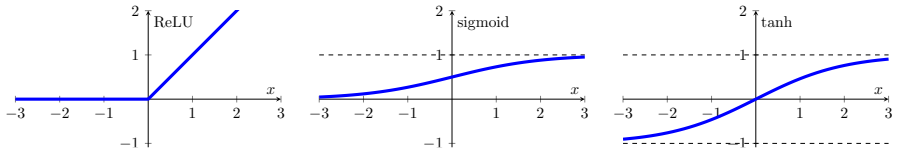
An Artificial Neural Network is a computational model biological inspired by the human brain processes. A basic unit of computation is the *neuron*. It receives input from some other neurons, or from an external source and computes an output. Each input has an associated weight w and a bias a. The neuron applies an *activation function* σ to the weighted sum of its inputs to compute the output. If we have n input $x_1, \dots, x_n$ we compute the output y as

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + a) \tag{1.3}$$

The activation function σ is a non-linear function. The purpose of the activation function is to introduce non-linearity into the output of a neuron. This is important because most real world data are non linear and we want neurons to learn these non linear representations.

There are different possible activation functions. In Figure 1.4 we plot three most common types.

The importance of non-linear activation functions becomes significant when one moves from the single-layer to multi-layer architectures.



ReLU: $\sigma(x) = \max(0, x)$ sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$ tanh: $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Figure 1.4: Three types of activation functions

1.3.1 Feedforward Neural Network

The feedforward neural network is the first and simplest type of artificial neural network devised. It contains multiple neurons arranged in layers. Neurons from adjacent *layers* have connections between them. All these connections have weights associated with them (see Figure 1.5).

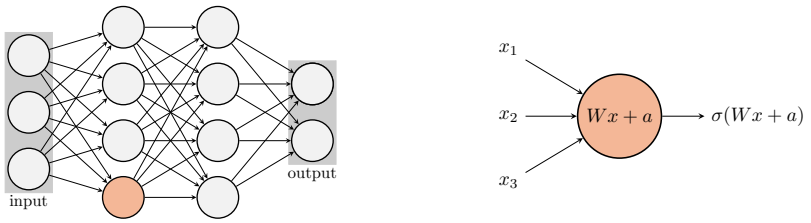


Figure 1.5: Feedforward neural network structure

These models are called feedforward because information flows from the input values x, through the hidden layers, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself.

Given a feedforward neural network $N : \mathbb{R}^n \to \mathbb{R}^m$ with $L > 2$ layers. The first layer is called input layer, the last is the output layer and all the intermediates are called hidden layers. We indicate with n_l the number of neurons in each layer $l = 1, \ldots, L$, with $n_1 = n$ and $n_L = m$. We call x^l the output variables at the layer l. Each layer has a matrix of weight $W^l \in \mathbb{R}^{n_{l-1} \times n_l}$ and a vector of bias $a^l \in \mathbb{R}^{n_l}$. We denote by h^l the function mapping an input x^{l-1} to the output x^l of the layer l

$$h^l(x^{l-1}) = \sigma(W^l x^{l-1} + a^l) \qquad i = 2, \ldots, n \tag{1.4}$$

Given and input $x \in \mathbb{R}^n$, the output $N(x) \in \mathbb{R}^m$ of the neural network is computed as the composition of the function h^l:

$$N(x) = (h^n \circ h^{n-1} \circ h^2)(x) \tag{1.5}$$

An interesting thing about feedforward networks with hidden layers is that it provides a universal approximation framework.

Theorem 1.1. (Universal Approximation [54, 27]) *A feedforward network with a linear output layer and at least one hidden layer can approximate any continuous function as accurately as desired.*

When we decide the structure of a specific model, i.e. the graph structure behind the network, we need to train the network in order to approximate our desired samples. Train a neural network is the optimization operation of tuning the weights and biases in order to minimize a cost function that is a measure difference between the actual NN output and the desired ones. The objective function is often referred to as a cost function or a loss function. There are many functions that could be used to estimate the error of a set of weights in a neural network. Standard example of loss functions are the followings:

- Mean Absolute Error (MAE)

$$C = \frac{1}{n} \sum_x |y - N(x)| \tag{1.6}$$

- Mean Square Error (MSE)

$$C = \frac{1}{n} \sum_x (y - N(x))^2 \tag{1.7}$$

- Root Mean Squared Error (RMSE)

$$C = \sqrt{\frac{1}{n} \sum_x (y - N(x))^2} \tag{1.8}$$

where n is the total number of training examples, x is an individual training example and $y = y(x)$ corresponds to the desired output.

Typically, a neural network model is trained using the stochastic gradient descent optimization algorithm and weights are updated using the backpropagation algorithm. The goal of backpropagation is to compute the partial derivatives $\frac{\partial C}{\partial w_{i,j}}$ and $\frac{\partial C}{\partial a_j}$ of the cost function C with respect to any weight $w_{i,j}$ of the matrix W or bias a_j of the vector a. For each training data the predicted value of the network is compared to the expected output, and an error is calculated using the selected cost function. This error is then propagated back within the whole network, one layer at a time, and the weights are updated according to the partial derivative value. One round of updating the network for the entire training dataset is called an epoch and a network may be trained for tens, hundreds or thousands epochs.

1.4 Summary

In this chapter we introduced notions and definitions related to optimization theory that we will use especially in Part II. We started by giving the definition of a multistage system, that is a system that describes a working cycle of a professional appliance. The underling structure of this system is a directed acyclic graph. Thus we introduced definition and notations related to the graph theory.

In Chapter 1.2 we gave the definition of an optimization problem and we defined a famous optimization problem over graphs: the traveling salesman problem. An application of this problem is described in Chapter 3.

We also defined a multi-objective optimization problem describing an important indicator to compare sets of optimal solution. The definitions of this section are preparatory for Chapter 4.

Finally in Chapter 1.3 we introduced the notation of a data-driven optimization tools: the Neural Networks. This tool and the analysis of its behavior is reported in Chapter 6.

2

Modeling: tools and problems

Hybrid systems are modeling tools used to describe processes which evolve according to dynamics and logic rules. These systems have been the subject of intensive study in the past few years. A particular emphasis has been placed on solving problems with safety specifications, which are described by giving a set of good states within which the controlled hybrid system should evolve.

A promising model for describing such systems is hybrid automata.

2.1 Hybrid automata

The notion of hybrid automata was introduced in [6] as a model and specification language for hybrid systems. The following definitions are standard notations of hybrid automata (see [6, 23] for more details).

Definition 2.1.1. *An* Hybrid Automaton $\mathcal{H} = \langle \mathbf{Z}, \mathbf{Z}', \mathcal{V}, \mathcal{E}, Inc, Dyn, Act, Reset \rangle$ *consists of the following components:*

- $\mathbf{Z} = (Z_1, \ldots, Z_k) \in \mathbb{R}^k$ *and* $\mathbf{Z}' = (Z'_1, \ldots, Z'_k) \in \mathbb{R}^k$ *are two vectors of variables;*
- $\langle \mathcal{V}, \mathcal{E} \rangle$ *is a finite directed graph: the vertices in* $\mathcal{V}$ *are called* locations *and the edges in* $\mathcal{E}$ *are called* transitions*;*
- *Each vertex* $v \in \mathcal{V}$ *is labeled by two formulas:*

  $Inv(v)[\mathbf{Z}]$ *that is an* invariant*, i.e. a constraint that the variables* $\mathbf{Z}$ *must respect during their evolution;*

  $Dyn(v)[\mathbf{Z}, \mathbf{Z}', t]$ *that is the* dynamic *law, in particular for each* $v \in \mathcal{V}$ *there is a continuous function* $f_v : \mathbb{R}^k \times \mathbb{R}_{\geq 0} \to \mathbb{R}^k$ *such as* $Dyn(v)[\mathbf{Z}, \mathbf{Z}', t]$ *is* $\mathbf{Z}' = f_v(\mathbf{Z}, t)$*;*

  The variable t *represents the time of the system and range over* $\mathbb{R}_{\geq 0}$*.*
- *Each edge* $e \in \mathcal{E}$ *is labeled by other two formulas:*

  $Act(e)[\mathbf{Z}']$ *is the* activation*, i.e. a constraint on the activation of the edge* e*;*

  $Reset(e)[\mathbf{Z}, \mathbf{Z}']$ *is the* reset *of the variables while we are using the edge* e*.*

Definition 2.1.2 (State)**.** *A* state σ *is a pair* $(v, \mathbf{x})$ *considering a location* $v \in \mathcal{V}$ *and a valuation* $\mathbf{x} \in \mathbb{R}^k$ *for the variables* $\mathbf{Z}$ *such as* $Inv(v)[\mathbf{x}]$ *holds.*

Definition 2.1.3 (Transition relations)**.** *Let* $\mathcal{H}$ *be a hybrid automaton. We have two type of transition relations:*

1. continuous transition relation $\rightarrow_t$ *between two states* $\sigma = (v, \mathbf{x})$ *and* $\delta = (v, \mathbf{y})$*is defined as follows:*
   $\sigma \rightarrow_t \delta \Leftrightarrow$ *there exists* $\mathbf{f} : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^k$ *continuous function such that* $\mathbf{x} = f(0)$, $\mathbf{y} = f(t)$ *and for each* $t' \in [0, t]$ *holds* $Inv(v)[\mathbf{f}(t')]$ *and* $Dyn(v)[\mathbf{x}, \mathbf{f}(t'), t']$
2. discrete transition relation $\rightarrow_e$ *between two states* $\sigma = (v, \mathbf{x})$ *and* $\delta = (w, \mathbf{y})$ *is defined as follows:*
   $\sigma \rightarrow_e \delta \Leftrightarrow e \in \mathcal{E}$ *such that* $e = (v, w)$ *and holds* $Inv(v)[\mathbf{x}]$, $Inv(w)[\mathbf{y}]$, $Act(e)[\mathbf{x}]$ *and* $Reset(e)[\mathbf{x}, \mathbf{y}]$

Definition 2.1.4 (Trajectory & Path)**.** *Let* $\mathcal{H}$ *be a hybrid automaton. A* trajectory *is a sequence* $(\sigma_j)_{j \in J}$ *of states such that:*

1. *for all* $j \in J \setminus \{0\}$ *there exists an edge* $e \in \mathcal{E}$ *such that* $\sigma_{j-1} \rightarrow_e \sigma_j$ *or there exists a time* $t > 0$ *such that* $\sigma_{j-1} \rightarrow_t \sigma_j$*;*
2. *for all* $j \in J \setminus \{0, 1\}$ *there exists* e *and* e' *in* $\mathcal{E}$ *such that* $\sigma_{j-2} \rightarrow_e \sigma_{j-1} \rightarrow_{e'} \sigma_j$ *or* $\sigma_{j-2} \rightarrow_e \sigma_{j-1} \rightarrow_t \sigma_j$ *or* $\sigma_{j-2} \rightarrow_t \sigma_{j-1} \rightarrow_e \sigma_j$*;*

A path *is a sequence of location* $v \in \mathcal{V}$ *for which there exists a corresponding trajectory of* $\mathcal{H}$*.*

Definition 2.1.5 (Reachability)**.** *Let* σ *and* δ *be two states of a hybrid automaton* $\mathcal{H}$*. The state* δ *is* reachable *from the state* σ *if there is a trajectory of* $\mathcal{H}$ *that starts in* σ *and ends in* δ*.*

Definition 2.1.6 (Reachability problem)**.** *Given an automaton* $\mathcal{H}$*, a set of starting states* Σ_S *and a set of ending states* Σ_E*, decide whether there exists a state* $\sigma \in \Sigma_S$ *from which a state in* Σ_E *is reachable.*

Reachability problem is a key part to the verification of a hybrid system. In particular the verification of safety properties is equivalent to the reachability question. *Safety* property informally requires that "something bad will never happen", means that the system should stay within a set of safe states, or equivalently, that the system should never reach a set of bad states.

- Safety problem requires that all trajectories that start in Σ_S do not reach a state of Σ_E
$$\forall k > 0 \ \forall (\sigma_j)_{j=1,\dots,k} (\sigma_1 \in \Sigma_S \rightarrow \sigma_k \notin \Sigma_E) \tag{2.1}$$
- Reachability problem is the negation of the safety problem, indeed the reachability problem asks if exists at least one trajectory that starts in Σ_S and ends in Σ_E
$$\exists k > 0 \ \exists (\sigma_j)_{j=1,\dots,k} (\sigma_1 \in \Sigma_S \wedge \sigma_k \in \Sigma_E) \tag{2.2}$$

We can set the safety property also in the positive way. Instead of avoid something bad we want to guarantee that we remain always into good states. Given a property φ, we want to verify that this property holds for a hybrid automaton $\mathcal{H}$; in other words, that φ remains true for all possible executions starting from a set Σ_S of starting states. If we call $\text{Reach}(\Sigma_S)$ the set of reachable states from Σ_S, then we only need to prove that $\text{Reach}(\varphi) \in Sat(\varphi)$, where $Sat(\varphi)$ is the set of states where φ is true.

2.2 Bernstein theory

Among methods for reachability analysis are those based on Bernstein expansion of polynomials. A Bernstein polynomial is a polynomial expressed in the Bernstein form, that is a linear combination of the Bernstein basis polynomials. As we will see in the following, the coefficients associated with these basis own interesting properties that can be exploited to bound the image of a polynomial over a unit box domain. Before giving the formal definition we can have an intuition of their usage looking at the plot of the Bernstein polynomial in the unitary interval (Figure 2.1). Depending on the degree we select, we are filling the interval $[0, 1]$ with a set of functions whose picks are a discretization of the interval and with the property that the sum of these functions is constantly 1. We can now imaging to manually play with this base of functions adding a set of coefficients to increase or decrease those picks in order to approximate a given function. The selected coefficients will directly describe the shape of the desired function.

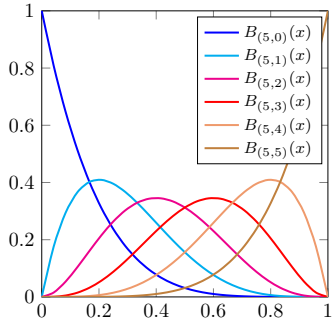


Figure 2.1: Bernstein polynomial base with degree 5.

We start now recalling all the definitions and properties of this theory, and we go further describing also some interesting generalizations. We developed all for function $f : \mathbb{R}^n \to \mathbb{R}$ but can be easily expanded to $f : \mathbb{R}^n \to \mathbb{R}^m$ just repeating all the steps for each $f_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \ldots, m$.

Lets consider a polynomial $p : \mathbb{R}^n \to \mathbb{R}$ expressed using the power basis as follows:

$$p(x) = \sum_{i \in I^n} a_i x^i \tag{2.3}$$

where the index $i = (i_1, \ldots, i_n)$ is a multi-index of size $n \in \mathbb{N}$ and x^i denotes the monomial $x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}$. We want to represent $p(x)$ using Bernstein basis.

2.2.1 Bernstein basis and properties

We need some additional definitions to continue. Given two multi-indices $i = (i_1, \ldots, i_n)$ and $j = (j_1, \ldots, j_n)$ we write $j \leq i$ if $j_k \leq i_k$ for $k = 1, \ldots, n$. We write i/j for the multi-index $(i_1/j_1, \ldots, i_n/j_n)$ and $\binom{i}{j}$ for the product $\binom{i_1}{j_1} \ldots \binom{i_n}{j_n}$. Finally we define the degree *deg* of p as the smallest multi-index such as $i \leq deg$ for all $i \in I^n$. In the following we will refer to $I^{deg} = \{i \in \mathbb{N}^n \mid i \leq deg\}$

A polynomial $p(x)$ can be represented using Bernstein basis as

$$p(x) = \sum_{i \in I^{deg}} b_i(p) B_{(deg,i)}(x) \tag{2.4}$$

where $b_i(p)$ are called *Bernstein coefficients* for a polynomial p and they are defined as

$$b_i(p) = \sum_{j \leq i} \frac{\binom{i}{j}}{\binom{deg}{j}} a_j \tag{2.5}$$

and the i-th *Bernstein polynomial* $B_{(deg,i)}$ of degree *deg* is

$$B_{(deg,i)}(x) = \beta_{(d_1,i_1)}(x_1) \ldots \beta_{(d_n,i_n)}(x_n) \tag{2.6}$$

with $deg = (d_1, \ldots, d_n)$ and

$$\beta_{(d_j,i_j)}(x) = \binom{d_j}{i_j} x^{i_j} (1-x)^{d_j - i_j} \tag{2.7}$$

Example 2.2.1. *Consider the polynomial $q(x) = x^2 + 1$, we want to represent it using Bernstein basis. We have $deg = 2$ and using Equation (2.5) we can compute the three Bernstein coefficients, $b_0(q), b_1(q)$ and $b_2(q)$ as follows:*

$$b_0(q) = \frac{\binom{0}{0}}{\binom{2}{0}} 1 = 1$$
$$b_1(q) = \frac{\binom{1}{0}}{\binom{2}{0}} 1 + \frac{\binom{1}{1}}{\binom{2}{1}} 0 = 1$$
$$b_2(q) = \frac{\binom{2}{0}}{\binom{2}{0}} 1 + \frac{\binom{2}{1}}{\binom{2}{1}} 0 + \frac{\binom{2}{2}}{\binom{2}{2}} 1 = 2$$

The final expression using Bernstein basis is

$$q(x) = 1\binom{2}{0}(1-x)^2 + 1\binom{2}{1}x(1-x) + 2\binom{2}{2}x^2 = (1-x)^2 + 2x(1-x) + 2x^2$$

For illustrative purpose lets consider also a bivariate polynomial $p(x_1, x_2) = \frac{1}{3}x_1^2 - \frac{1}{2}x_2 + \frac{1}{4}x_1x_2 + \frac{1}{2}$. We have $deg = (2, 1)$ and 6 Bernstein coefficients to compute. We

just calculate $b_{(1,1)}$ *and* $b_{(1,0)}$*:*

$$b_{(1,1)} = \frac{\binom{(1,1)}{(0,0)}}{\binom{(2,1)}{(0,0)}}\frac{1}{2} - \frac{\binom{(1,1)}{(0,1)}}{\binom{(2,1)}{(0,1)}}\frac{1}{2} + \frac{\binom{(1,1)}{(1,0)}}{\binom{(2,1)}{(1,0)}}0 + \frac{\binom{(1,1)}{(1,1)}}{\binom{(2,1)}{(1,1)}}\frac{1}{4} = \frac{1}{8}$$
$$b_{(1,0)} = \frac{\binom{(1,0)}{(0,0)}}{\binom{(2,1)}{(0,0)}}\frac{1}{2} + \frac{\binom{(1,0)}{(1,0)}}{\binom{(2,1)}{(1,0)}}0 = \frac{1}{2}$$

Bernstein coefficients present three interesting property.

Property 2.1. (Sharpness) *For all* $i \in \mathcal{V}_{deg}$

$$b_i(p) = p(i/deg) \tag{2.8}$$

with $\mathcal{V}_{deg}$ *the set of vertices of the hyperrectangle* $[0, d_1] \times \ldots [0, d_n]$ *with* $deg = (d_1, \ldots, d_n)$*.*

.

Property 2.2. (Range Enclosing)

$$\min_{i \in I^{deg}} b_i(p) \leq p(x) \leq \max_{i \in I^{deg}} b_i(p) \tag{2.9}$$

for all $x \in [0, 1]^n$

These two properties together states that we can bound a polynomial function in the unitary domain and the vertices of the box enclosing, match exactly the values of the polynomial at some points. Property 2.2 is true on the unitary hyperbox, but can be easily extended to a hyperrectangle composing the polynomial with an affine transformation between the hyperrectangle and the unitary box [34].

The last important property allows to enclose the input/output polynomial relation with a convex structure.

Definition 2.2.1. *Let* $V = \{v_1, \ldots, v_l\}$ *be a sequence of points in* $\mathbb{R}^n$*. The* convex hull *of* V *is the set*

$$Convex_{hull}(V) = \{y \in \mathbb{R}^n \mid y = \sum_{i=1}^{l} \alpha_i v_i, \sum_{i=1}^{l} \alpha_i = 1,\ \alpha_i \geq 0\}$$

Property 2.3. (Convex hull) *Let* $p \in [0, 1]^n \to \mathbb{R}^m$ *be a polynomial of degree deg, we have*

$$\begin{pmatrix} x \\ p(x) \end{pmatrix} \subseteq Convex_{hull}\Big(\Big\{ \begin{pmatrix} v_i \\ b_i(p) \end{pmatrix} \mid i \in I^{deg} \Big\}\Big) \tag{2.10}$$

were $v_i := \frac{i}{deg}$ *and* $(v_i, b_i(p))$ *is called* control point *associated with the* i*-th Bernstein coefficient.*

Example 2.2.2. *We consider again the two polynomial in Example 2.2.1. We start with* $q(x) = x^2 + 1$ *and we can apply Property 2.1 for computing* $b_0(q) = \frac{0}{2}^2 + 1 = 1$ *and* $b_2(q) = \frac{2}{2}^2 + 1 = 2$. *Thanks to Property 2.2 these values bound exactly the output of the polynomial function in* $[0, 1]$. *In addition, if we consider the 3 control points* $\{(0, 1), (0.5, 1), (1, 2)\}$ *and we link them with a triangle we can enclose the polynomial* $q(x)$ *as we can see in Figure 2.2*

As reguard $p(x_1, x_2) = \frac{1}{3}x_1^2 - \frac{1}{2}x_2 + \frac{1}{4}x_1x_2 + \frac{1}{2}$, *with the sharpness property we can obtain*

$$
\begin{aligned}
b_{(0,0)} &= \frac{1}{2} \\
b_{(2,0)} &= \frac{1}{3}\left(\frac{2}{2}\right)^2 + \frac{1}{2} = \frac{5}{6} \\
b_{(0,1)} &= -\frac{1}{2} \cdot \frac{1}{1} + \frac{1}{2} = 0 \\
b_{(2,1)} &= \frac{1}{3}\left(\frac{2}{2}\right)^2 - \frac{1}{2} \cdot \frac{1}{1} + \frac{1}{4} \cdot \frac{2}{2} \cdot \frac{1}{1} + \frac{1}{2} = \frac{7}{12}
\end{aligned}
$$

Knowing also the remaining 2 coefficients from the previous example we can bound the output $p(x_1, x_2) \in [0, \frac{5}{6}]$ *when* $x_1, x_2 \in [0, 1]$ *as we can see in Figure 2.3. Moreover, we can link all the control points and obtain a convex hull that enclose the surface.*

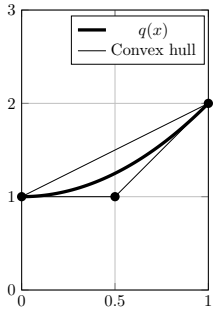


Figure 2.2: Polynomial function $q(x)$ in $[0, 1]$ enclosed by the convex hull of its control points.

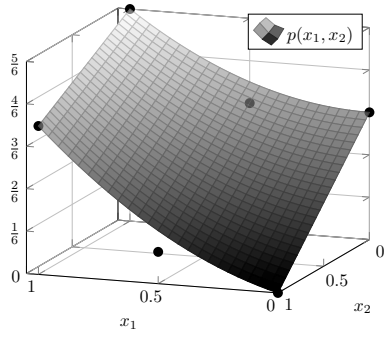


Figure 2.3: Polynomial function $p(x_1, x_2)$ with $x_1, x_2 \in [0, 1]$. The 6 black dots are the control points.

All the properties described so far can be apply for polynomial functions on unitary boxes or in general on hyperrectangular domains, what if we consider more general domains? Are these properties still valid? What about a rational function? We discuss here two generalization: first the generalization over simplices domains that allows to extended the Bernstein theory over any convex domain and then the extension of the Bernstein properties for rational functions.

2.2.2 Simplicial Bernstein generalization

In order to extend the Bernstein properties over different domains, keeping the same properties, we need to change the system of coordinates searching for non-negative coordinates. Moreover the coordinates definition should be easy enough to be generalized in n-dimension and we should also be able to compute the new Bernstein coefficients to take advantage of the properties. In [112, 32] they define an easy extension for general polytope, i.e. a bounded convex set defined by a set of linear inequalities. Unfortunately this coordinates are not positive in the total convex hull. An alternative definition is given in [69]. Here the coordinates are positive on the convex hull but the construction is not easy to be generalized in n-dimensions. In both cases the estimation of the Bernstein coefficients is not explained. The majority of the generalized barycentric coordinates are developed just for 3D and used for graphical purposes [94, 72, 60, 44, 115]. The easiest generalization that keeps the desired property is the Simplicial Bernstein generalization, i.e. the Bernstein base over a simplex [41].

Definition 2.2.2. *Let $V = \{v_1, \dots, v_m\}$ be a sequence of points in $\mathbb{R}^n$. V is* affinely independent *if*

$$\forall x \in \mathbb{R}^n \ \exists! \xi(x) \in \mathbb{R}^m : x = \sum_{i=1}^{m} \xi_i(x) v_i, \quad \sum_{i=1}^{m} \xi_i(x) = 1$$

The functions $\xi_i : \mathbb{R}^n \to \mathbb{R}$ so defined are called barycentric coordinates *and they are non-negative on the convex hull of the points in V.*

We denote by $e_1, \dots, e_n$ the canonical basis of $\mathbb{R}^n$ and by e_0 the zero vector in $\mathbb{R}^n$. For $n \geq 0$ the *standard simplex* Δ_n is the set $\Delta_n \subset \mathbb{R}^n$ defined by

$$\Delta_n = \{\sum_{i=1}^{n} \xi_i(x) e_i \mid \sum_{i=1}^{n} \xi_i(x) = 1, \xi_i \geq 0\} \tag{2.11}$$

If $x = (x_1, \dots, x_n) \in \Delta$ then the barycentric coordinates with respect to Δ are $\xi(x) = (\xi_0(x), \dots, \xi_n(x)) = (1 - |x|, x_1, \dots, x_n)$ with $|x| = x_1 + \cdots + x_n$. Another way to describe Δ_n is to say that it is the convex hull of the standard basis $\{e_0, \dots, e_n\}$. We can define a general simplex $S \subset \mathbb{R}^n$ as the convex hull of $n+1$ vectors $\{v_0, \dots, v_n\}$. The simplex S is non-degenerate if $\{v_0, \dots, v_n\}$ are affinely independent.

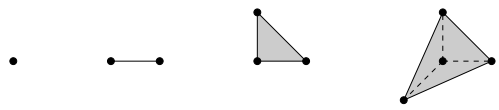


Figure 2.4: Standard simplex Δ_n with $n = 0, 1, 2, 3$.

We want to define a Bernstein polynomial over a standard simplex. Lets first of all observe the following property of the Bernstein polynomials defined in Equation

(2.7). Given an $x \in \mathbb{R}$ and $k \in \mathbb{N}$ we have

$$1 = 1^k = (x + (1-x))^k = \sum_{i=1}^{k} \binom{k}{i} x^i (1-x)^{k-i}$$
$$= \sum_{i=1}^{k} \beta_{(k,i)}(x)$$

This observation gives us an idea to define a generalize multinomial expression. Given a vector $\xi = \{\xi_1, \ldots, \xi_m\}$ with $\sum_i \xi_i = 1$ we introduce the following notation for multinomial expansion

$$(\xi_1 + \cdots + \xi_m)^k = \sum_{\alpha_1 + \cdots + \alpha_m = k} \binom{k}{\alpha_1, \ldots, \alpha_m} \xi_1^{\alpha_1} \ldots \xi_m^{\alpha_m}$$
$$= \sum_{|\alpha| = k} \binom{|\alpha|}{\alpha} \xi^\alpha$$

with $\alpha \in \mathbb{N}^m$ and

$$\binom{k}{\alpha} = \binom{k}{\alpha_1, \ldots, \alpha_m} = \frac{k!}{\alpha_1! \ldots \alpha_m!}$$

Definition 2.2.3. *Given $V = \{v_1, \ldots, v_m\}$ and a polynomial*

$$p : Convex_{hull}(\{v_1, \ldots, v_m\}) \to \mathbb{R},$$

the generalized Bernstein polynomial *of degree k is*

$$B_V^{(k)} p = \sum_{|i| = k} b_i^{(k)}(p, V) B_{i,V}^{(k)}$$

with $\xi = (\xi_1, \ldots, \xi_m)$ the generalized barycentric coordinate with respect to V, and

$$B_{i,V}^{(k)}(x) = \binom{|i|}{i} \xi^i(x)$$

Given a general simplex S of vertices $\{v_0, \ldots, v_n\}$ with $v_i \in \mathbb{R}^n$ we can map affinely each point of S into the standard simplex Δ and vice-versa. In particular given $x \in \Delta$ we obtain $y \in S$ as

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = T(x) := V \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + v_0 \tag{2.12}$$

with

$$V = \begin{pmatrix} v_1(1) - v_0(1) & \ldots & v_n(1) - v_0(1) \\ \vdots & & \vdots \\ v_1(n) - v_0(n) & \ldots & v_n(n) - v_0(n) \end{pmatrix}$$

W.l.o.g., as the case of the unitary box, we develop the theory just for the standard simplex Δ.

We introduce additional notations. Given $\alpha = (\alpha_0, \ldots, \alpha_n) \in \mathbb{N}^{n+1}$ we indicate with $\hat{\alpha} = (\alpha_1, \ldots, \alpha_n)$. For $\hat{\alpha}, \hat{\beta} \in \mathbb{N}^n$ with $\hat{\alpha} \leq \hat{\beta}$, we use

$$\binom{\hat{\alpha}}{\hat{\beta}} := \prod_{i=1}^{n} \binom{\alpha_i}{\beta_i} \tag{2.13}$$

and if $|\hat{\alpha}| < k$ then

$$\binom{k}{\hat{\alpha}} := \binom{k}{\alpha_1, \ldots, \alpha_n} = \frac{k!}{\alpha_1! \ldots \alpha_n! (k - |\hat{\alpha}|)!} \tag{2.14}$$

Definition 2.2.4. ([51]) *Given a (multivariate) polynomial of degree l*

$$p(x) = \sum_{|\hat{\beta}| \leq l} a_{\hat{\beta}} x^{\hat{\beta}}$$

the Simplicial Bernstein generalization *of p of degree $k \geq l$ on Δ is*

$$p(x) = \sum_{|\alpha| = k} b_{\alpha}^{(k)}(p, \Delta) B_{\alpha}^{(k)} \tag{2.15}$$

where

$$B_{\alpha}^{(k)} = \binom{k}{\alpha} \xi^{\alpha} \tag{2.16}$$

and

$$b_{\alpha}^{(k)}(p, \Delta) = \sum_{\hat{\beta} \leq \hat{\alpha}} \frac{\binom{\hat{\alpha}}{\hat{\beta}}}{\binom{k}{\hat{\beta}}} a_{\hat{\beta}}, \quad |\hat{\alpha}| \leq k \tag{2.17}$$

Definition 2.2.5. *Given the Bernstein coefficients of degree $k \geq l$ of a (multivariate) polynomial p of degree l over a general simplex S, the* control point *associated with the α-th Bernstein coefficient $b_{\alpha}^{(k)}(p, S)$ is the point $(v_{\alpha}^{(k)}(p, S), b_{\alpha}^{(k)}(p, S))$ with*

$$v_{\alpha}^{(k)}(p, S) := \frac{\alpha_0 v_0 + \cdots + \alpha_n v_n}{k} \in \mathbb{R}^n, \quad |\alpha| = k \tag{2.18}$$

Example 2.2.3. *Lets consider again the polynomial $p(x_1, x_2) = \frac{1}{3}x_1^2 - \frac{1}{2}x_2 + \frac{1}{4}x_1x_2 + \frac{1}{2}$ and lets define it over the standard simplex Δ_2. $deg = (2, 1)$ so we can define the Simplicial Bernstein generalization of degree 3. Lets compute just $b_{(1,1,1)}^{(3)}(p, \Delta)$:*

$$b_{(1,1,1)}^{(3)}(p, \Delta) = \frac{\binom{(1,1)}{(0,0)}}{\binom{3}{(0,0)}} \frac{1}{2} - \frac{\binom{(1,1)}{(0,1)}}{\binom{3}{(0,1)}} \frac{1}{2} + \frac{\binom{(1,1)}{(1,0)}}{\binom{3}{(1,0)}} 0 + \frac{\binom{(1,1)}{(1,1)}}{\binom{3}{(1,1)}} \frac{1}{4} = \frac{1}{2} - \frac{1}{6} + \frac{1}{24} = \frac{3}{8}$$

The associated control point is $(\frac{1}{3}, \frac{1}{3}, \frac{3}{8})$

With this generalization we still have the same main properties.

Property 2.4. (Sharpness [72]) *Keeping the same notations, we have*

$$b^{(k)}_{ke_i}(p,\Delta) = p(v_{ke_i}) \quad \forall i \in \{0,\dots,n\} \tag{2.19}$$

Property 2.5. (Range Enclosing [51]) *The output of a polynomial p over S is contained within the maximum and the minimum Simplicial Bernstein coefficients*

$$min_{|\alpha|=k} b^{(k)}_\alpha(p,S) \leq p(S) \leq max_{|\alpha|=k} b^{(k)}_\alpha(p,S) \tag{2.20}$$

Property 2.6. (Convex Hull [51]) *The polynomial p over S is contained within the convex hull of the control points*

$$\begin{pmatrix} x \\ p(x) \end{pmatrix} \subseteq Convex_{hull}\Big(\Big\{ \begin{pmatrix} v^{(k)}_\alpha(p,S) \\ b^{(k)}_\alpha(p,S) \end{pmatrix} \mid |\alpha| = k \Big\}\Big) \tag{2.21}$$

2.2.3 Bernstein for rational function

The last generalization we want to consider is the Bernstein theory for rational function so the ratio between two polynomial functions. We consider $f = \frac{p}{q}$ rational function with p, q polynomial functions. Let us compute the Bernstein coefficients $b^{(k)}_\alpha(p)$ and $b^{(k)}_\alpha(q)$, over a box X or the standard simplex Δ. Assume that all Bernstein coefficients $b^{(k)}_\alpha(q)$ have the same sign and are non-zero (this implies for the range enclosing property that $q(x) \neq 0$, for all $x \in X$). Then in [82] they prove that holds Property 2.2 for the rational function so

$$\min_{|\alpha|=k} \frac{b^{(k)}_\alpha(p)}{b^{(k)}_\alpha(q)} \leq f(x) \leq \max_{|\alpha|=k} \frac{b^{(k)}_\alpha(p)}{b^{(k)}_\alpha(q)} \tag{2.22}$$

with $k \geq$ the degree of p and q.

Unfortunately in [82] they also show that the convex hull property does not hold for rational functions and they create also a counterexample.

Generalize the Bernstein theory for rational functions in not as easy as the simplex generalization. The convex hull property does not hold anymore and the range enclosing property holds under a strict condition for the denominator. Indeed, to apply the range enclosing property we need to guarantee that the Bernstein coefficients of the denominator have the same sign. As we can see in Example 2.2.4, we can have negative Bernstein coefficients even with a positive function.

Example 2.2.4. *Lets consider the rational function $f(x) = \frac{p(x)}{q(x)} = \frac{x^2-1}{x^2+1}$ over $[-2,1]$. To compute the Bernstein coefficients we transform the domain into the unitary interval using the transformation $T : [0,1] \to [-2,1]$ defined as*

$$T(u) = 3u - 2$$

$p(T(u))$ and $q(T(u))$ are polynomial functions on $[0,1]$. Computing the Bernstein coefficients we have

$$\begin{aligned} b_0(p) &= 3 & b_0(q) &= 5 \\ b_1(p) &= -3 & b_1(q) &= -1 \\ b_2(p) &= 0 & b_2(q) &= 2 \end{aligned}$$

Then we can take the ration and we obtain that the rational function is wrongly bounded by the interval $[0, 3]$ *(see Figure 2.5). The denominator is a positive function but the second Bernstein coefficient is negative and the others are positive. This happens because the point* $x = 0$ *where we have the minimum of the function* q *is not included in the discretization of the interval (see Figure 2.6).*

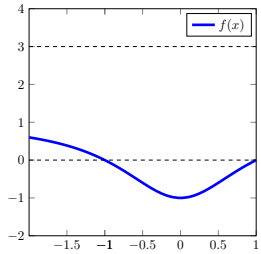


Figure 2.5: Rational function $f(x)$ wrongly bounded in $[0, 3]$.

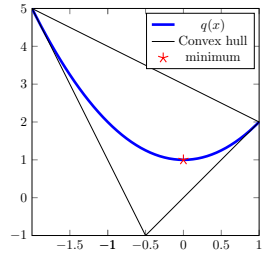


Figure 2.6: Convex hull for $q(x) > 0$ with a negative Bernstein coefficient.

We need to address the problem of deciding whether a given polynomial or rational function in n-variables has a *positive certificate*, i.e. all its Bernstein coefficients are positive. In [106] they prove Theorem 2.1 on a simplex, but the same holds also for hyperrectangle.

Theorem 2.1. *Let* p *be a polynomial of degree* $d \leq k$ *over a simplex* Δ*. Then*

$$min_{x\in\Delta} p(x) = min_{|\alpha|=k} b_\alpha^{(k)}(p) \tag{2.23}$$

if and only if

$$min_{|\alpha|=k} b_\alpha^{(k)}(p) = b_{\alpha^*}^{(k)}(p) \tag{2.24}$$

for some $\alpha^* = ke_i$ *with* $i \in \{0, \ldots, n\}$

These results suggest us that the Bernstein coefficients of a positive function are positive only if we are able to select the minimum in our control points. In Example 2.2.5 we apply this technique for the polynomial expression in Example 2.2.4.

Example 2.2.5. *Consider again the rational function* $f(x) = \frac{p(x)}{q(x)} = \frac{x^2-1}{x^2+1}$ *over* $[-2, 1]$*. The denominator* $q(x)$ *has the minimum in* $x = 0$ *so we split the domain in* $X_1 = [-2, 0]$ *and* $X_2 = [0, 1]$*. The Bernstein coefficients for* $q(x)$ *are*

$$\begin{aligned} &b_0(q, X_1) = 5; && b_1(q, X_1) = 1; && b_2(q, X_1) = 1; \\ &b_0(q, X_2) = 1; && b_1(q, X_2) = 1; && b_2(q, X_2) = 2. \end{aligned}$$

They are all positive and we obtain two convex hull as in Figure 2.8. We can finally compute the ratio of the Bernstein coefficients for the two domains

$$\begin{aligned} &b_0(f, X_1) = \frac{3}{5}; && b_1(f, X_1) = -1; && b_2(f, X_1) = -1; \\ &b_0(f, X_2) = -1; && b_1(f, X_2) = -1; && b_2(f, X_2) = 0. \end{aligned}$$

and bound correctly the function as in Figure 2.7.

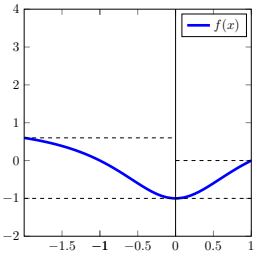


Figure 2.7: Rational function $f(x)$ correctly bounded in $[-1, \frac{3}{5}]$ when $x \in [-2, 0$ and $[-1, 0]$ when $x \in [0, 1]$.

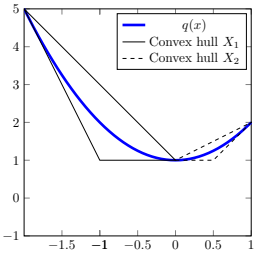


Figure 2.8: Two convex hull for q with positive Bernstein coefficients.

Matrix method to compute Bernstein coefficients

To conclude with the Bernstein theory we want to describe a matrix method to compute the coefficients. It is well known that we can compute the Bernstein coefficients over a hyperrectangle using a matrix product. This method was introduced by Garloff in [47]. In [104], Garloff and Titi expand the matrix calculation for simplicial Bernstein expressions. The idea behind is the same, so we describe it for the simplex case and the hyperrectangle case can be considered a subcase, see [105] for additional details.

Let $q : \mathbb{R}^n \to \mathbb{R}$ be a polynomial defined over a simplex $S = \{v_0, \ldots, v_n\}$. Using the transformation 2.12 we define the polynomial $p(x) = q(T(x))$ over the standard simplex Δ.

$$p(x) = \sum_{|\hat{\alpha}| \leq l} a_{\hat{\alpha}} x^{\hat{\alpha}} \tag{2.25}$$

The Bernstein polynomial has a degree k such as $l \leq k$. We arranged the coefficients of p in an $(k+1) \times k^*$ matrix A with $k^* := (k+1)^{n-1}$

$$A := \begin{pmatrix} a_{0,0,\ldots,0} & a_{0,1,\ldots,0} & \cdots & a_{0,k,\ldots,0} & \cdots & a_{0,k\ldots,k} \\ a_{1,0,\ldots,0} & a_{1,1,\ldots,0} & \cdots & a_{1,k,\ldots,0} & \cdots & a_{1,k\ldots,k} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{k,0,\ldots,0} & a_{k,1,\ldots,0} & \cdots & a_{k,k,\ldots,0} & \cdots & a_{k,k\ldots,k} \end{pmatrix}$$

with $a_{\hat{\alpha}} = 0$ if $|\hat{\alpha}| > l$. We also define the lower triangular Pascal matrix P, i.e.

$$(P_k)_{i,j} = \begin{cases} \binom{i-1}{j-i} & j \leq i \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

We can compute $P_k = \prod_{h=1}^{k} K_h$ from the matrices K_h, $h = 1, \ldots, k$ defined as

$$(K_h)_{i,j} = \begin{cases} 1 & j \leq i \\ 1 & i = j+1,\ j \geq k-h+1 \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

We introduce the *cyclic ordering* of the entries of a matrix denoted with the superscript c, i.e. the order of the indices of the entries of the matrix under a cyclic permutation. This means that the index in the first position is replaced by the index in the second one, the index in i-th position by the one in the $i+1$-th position and the index in the n.th position by the one in the first position. In two dimensions the cyclic ordering is the transposition. We put $C_0(\Delta) := A$ and define for $t = 1, \ldots, n$

$$C_t := (P_k C_{t-1})^c \tag{2.28}$$

The Bernstein coefficients of the polynomial p over Δ can be obtained for the entries of the matrix C_n considering just $(C_n)_{\hat{\alpha}}$ with $|\hat{\alpha}| \leq k$.

$$b_\alpha^{(k)} = b_{(k-|\hat{\alpha}|,\hat{\alpha})}^{(k)} = (C_n)_{\hat{\alpha}} \tag{2.29}$$

Finally we compute v_α as

$$v_\alpha^{(k)} = T(\frac{\alpha_0 e_0 + \cdots + \alpha_n e_n}{k}) = T(\frac{\hat{\alpha}}{k}) \tag{2.30}$$

2.3 Summary

In this chapter we introduced notions and definitions related to the theory of hybrid systems that we will use in Part III. In Chapter 4.2.1 we defined the hybrid automaton formalism and we defined the reachability problem. These two concepts are important to understand the second approach for the thawing case proposed in Chapter 4.2. Moreover the reachability problem is a basic concept for the whole Part III.

To solve the reachability problem in this thesis we decided to exploit the technique based on Bernstein theory. In Chapter 2.2 we collected all definitions, properties and theorems of Bernstein theory both for rectangular and simplex domains. We described the polynomial case that is used particularly in Chapter 5 and the rational case that is functional for Chapter 6.

II

Optimization of multistage systems

3

Scheduling of recipes for a professional oven application

The soaring demand for productivity and flexibility in food catering sectors, canteens or food retail, has increased and highlighted the value of cooked-chilled and cooked-frozen food products. Anyone who has eaten at a top restaurant or hotel, at a banquet or reception, or on an airplane or ship is likely to have eaten a Cook & Chill meal.

Cook & Chill and Cook & Freeze are well known food process techniques used to prepare meals in advance and store them safely till the right serving time maintaining a consistent quality [39, 71]. In the Cook & Chill the meals or meal components are fully cooked, then cooled by controlled chilling, e.g. blast chilling, to reduce the temperature in the center of the food from about $60°C$ to $10°C$ in less than two hours; subsequently stored at a temperature above freezing point ($4°C$) and finally reheated/regenerated rapidly before the service. In the Cook & Freeze the meals or meal components should be maintained at or below $-18°C$ [22].

Different benefits promote the use of these methods. Above all the production and preparation of a meal are spatially and temporary independent of the service and consumption. The kitchen is not forced to follow the service request and the lunch/dinner hours but can distribute its workflow during the day. This allows to plan the production ahead and systematically, to reduce product and time waste and to avoid repetitive cook and serve operations.

Nowadays the scheduling is made up according to the need of the chef, without following any technical criteria, e.g. the energy and water consumption or the total time. We want to study an automatic function able to sort a list of cooking processes with respect to energy consumption. As reported in [80, 86] the energy consumption is a relevant cost factor in professional kitchen. Food service facilities have an average energy use almost three times higher than other commercial activities. Therefore we select the energy as the technical criteria to optimize the scheduling.

The automatic function considers a list of cooking processes, each one characterized by a set of parameters as temperature, humidity and duration, and it outputs a reordered sequence that minimizes the total energy consumption. This result is achieved taking into account only the energy consumption during the transient phases between consecutive cooking processes. The total energy consumption during

the stationary phase of each cooking process can be considered invariant among the possible reorders. We start formalizing the problem and the variables involved, then we characterize the energy function and finally we describe the implemented solution.

3.1 Scheduling problem definition

The automatic function that we want to define has to solve the following problem.

Definition 3.1.1. (Problem statement) *Given a list of m cooking processes we want to find a sequence π that minimizes the total energy, i.e. the energy consumption needed to heat up the oven, execute all the cooking processes following π and cool down again the appliance.*

A cooking process C is a multistage system with $n > 1$ stages. We call the first stage *transient* stage. It should be performed with the cavity empty and it is intended to prepare the oven with the right temperature and humidity. The transient stage ends when the oven reaches the desired values. It could be a *warm up* phase if we are rising the temperature or a *cooling down* phase if we need to decrease the temperature inside the cavity of the oven. The other stages of C depend on the specific recipe and they are characterized by a temperature value, a humidity percentage and a duration. For each cooking process C with n stages we can define a vector $T_C = (T_1^C, \dots, T_n^C)$ of temperatures and a vector $H_C = (H_1^C, \dots, H_n^C)$ of humidity levels.

We start considering just two cooking processes C_1 and C_2 with n_1 and n_2 stages, respectively. If we select the sequence $(1, 2)$ we will have a total energy $E_{(1,2)}$ that depends on the energy needed for the two cooking processes (E_{C_1}, E_{C_2}), the energy needed to warm up the oven before the first cooking process (E_{WU-C_1}), the energy required to pass from the ending temperature of C_1 to the first temperature of C_2 ($E_{C_1-C_2}$), and finally the energy to cool down the oven at the end of the sequence (E_{CD-C_2}). So in the end we have the following quantity

$$E_{(1,2)} = E_{WU-C_1} + E_{C_1} + E_{C_1-C_2} + E_{C_2} + E_{CD-C_2}$$

If we select the sequence $(2, 1)$ we obtain

$$E_{(2,1)} = E_{WU-C_2} + E_{C_2} + E_{C_2-C_1} + E_{C_1} + E_{CD-C_1}$$

We can assume that the energy consumption when the foodstuff is inside the cavity does not depend on the order but is related just to the specific recipe and the food properties. What we can change is the energy during the transient stages between consecutive cooking processes (warm-up or cooling-down stages). In other words, we can neglect the terms E_{C_2} and E_{C_1} and work on the remaining ones.

Given a set of cooking processes $\mathcal{J} = \{C_1, \dots, C_m\}$, we need to define an energy function $f : \mathcal{J} \times \mathcal{J} \to \mathbb{R}$ such that $f(C_i, C_j)$ defines the energy to pass from the last stage of the cooking process C_i to the first stage of the cooking process C_j. In the example above $E_{C_1-C_2} = f(C_1, C_2)$.

We need to define a dummy cooking process C_0 with a single stage characterized by the environment temperature T_0 and a humidity 0%. This correspond to the

starting/ending condition, when the oven is off. With this definition we have, for example, $E_{WU-C_1} = f(C_0, C_1)$.

We can now define the problem in Definition 3.1 as a scheduling problem where each cooking process is a job and the energy function is a switching cost to pass from one job to the next one.

Definition 3.1.2. (Scheduling problem) *Given a set $\mathcal{J} = \{C_0, \ldots, C_m\}$, with m jobs $(C_i)_{i\in\{1,\ldots,m\}}$ and a starting/ending condition C_0, and given a weight function $f : \mathcal{J} \times \mathcal{J} \to \mathbb{R}$, select a permutation π of $\{1, \ldots, m\}$ in order to minimize the total cost*

$$cost = f(C_0, C_{\pi(1)}) + \sum_{i=1}^{m-1} f(C_{\pi(i)}, C_{\pi(i+1)}) + f(C_{\pi(m)}, C_0)$$

Before addressing the solution of the problem we need to define the energy function $f(C_i, C_j)$.

3.2 Characterization of the energy function

We already said that we want to define this energy considering just the last stage of a cooking process C_i characterized by a set of temperature and the first stage of the consecutive cooking process C_j. The energy $f(C_i, C_j)$ is the energy to pass from temperature $T_{n_i}^{C_i}$ and humidity $H_{n_i}^{C_i}$ to temperature $T_1^{C_j}$ and humidity $H_1^{C_j}$. Basically the function e represents the energy consumption to move from a stage defined by (T, H) to a stage with values (T', H'). Instead of defining a function of four variables (2 temperatures and 2 humidity levels) we decide to use only two variables: $\bar{T}$ and ΔH. $\bar{T}$ represents the change in temperature from T to T' weighted with a corrective factor and ΔH is the simple variation of humidity from H to H'.

Use the energy balance is a traditional method to evaluate the energy efficiency of a process but it does not provide information on the degradation of energy during a process. For our purpose the concept of exergy is more indicated to describe the difference between sequences. The concept of *exergy* [57] is the maximum useful work possible during a process. In a heating process this depends on the temperature at which heat is available and the temperature level at which the rejected heat can be disposed. We use this idea and the definition of exergy as an inspiration to modify the temperature variation with the Carnot coefficient

$$\bar{T} = \Delta T(1 - \frac{T_0}{T}) \tag{3.1}$$

where $\Delta T = (T' - T)$ and T_0 is the environment temperature.

We have to describe the function f as a function of $\bar{T}$ and ΔH. The function f must be characterized for the specific appliance that we want to consider. The analytic expression depends on the dimensions of the oven, the heating elements and the insulation. The expression could become very complex and must be recalculated for each appliance. For this reason we decide to fit some experimental tests to create a surrogate polynomial model p_f that approximates the energy consumption. All the

information are obtained measuring the energy consumption with some idle tests, so by increasing or decreasing the temperature and humidity of the oven with the cavity empty.

We design the tests in order to cover a realistic portion of the domain. During the test campaign we noticed that the behavior of the warm up stages, where $\bar{T} > 0$, is functionally different from the behavior during cool down stages, with $\bar{T} < 0$. For this reason we decide to approximate the energy consumption with two second degree polynomial function : one for $\bar{T} \geq 0$ (E_{pos}) and one for $\bar{T} \leq 0$ (E_{neg}).

Our final function p_f is a piecewise polynomial function

$$p_f(\bar{T}, \Delta H) = \begin{cases} E_{pos}(\bar{T}, \Delta H) & \bar{T} > 0 \\ E_{neg}(\bar{T}, \Delta H) & \bar{T} \leq 0 \end{cases} \tag{3.2}$$

where

$$E_{neg}(\bar{T}, \Delta H) = n_{1,1}\bar{T}\Delta H + n_{2,0}\bar{T}^2 + n_{0,2}\Delta H^2 + n_{1,0}\bar{T} + n_{0,1}\Delta H + n_{0,0}$$
$$E_{pos}(\bar{T}, \Delta H) = p_{1,1}\bar{T}\Delta H + p_{2,0}\bar{T}^2 + p_{0,2}\Delta H^2 + p_{1,0}\bar{T} + p_{0,1}\Delta H + p_{0,0}$$

and the coefficients $p_{i,j}$ and $n_{i,j}$ are obtained by fitting the experimental data.

3.3 Asymmetric traveling salesman problem

In order to solve the scheduling problem in Definition 3.1.2 we reformulate it as an asymmetric traveling salesman problem (ATSP). Let us consider a complete graph K_n with $n = m+1$ as in Figure 3.1. Each vertex is a job in $\mathcal{J}$. For each edge $e = (i, j)$ we define the same cost function $w_{i,j} = f(C_i, C_j)$. We want to find a minimum weight path that starts in C_0, visits all the nodes and comes back to C_0, so a Hamiltonian cycle.

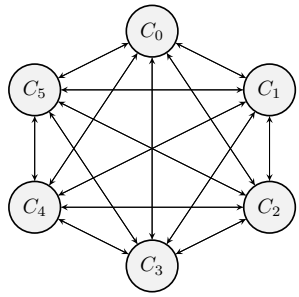


Figure 3.1: Complete graph K_6

Definition 3.3.1. (ATSP) *Given a direct graph $G = (V, E)$ with n nodes and a weight function $w : V \times V \to \mathbb{R}$, find a Hamiltonian cycle H such as $w_H = \sum_{e \in H} w_e$ is minimum.*

In literature there are bunch of algorithms to solve the ATSP. In order to implement a light algorithm that has to be included inside a real machine, we decide to solve the problem using a greedy approach.

We implement the classical Karg-Thompson heuristic [62]. This heuristic is a greedy algorithm that strongly depends on the initial order. The pseudocode is described in Algorithm 3.1

Algorithm 3.1 Karg-Thompson heuristic

Input: A complete graph K_{n+1} and a weight function $w : V \times V \to \mathbb{R}$
Output: A vector V of reordered indices

Select a permutation π of $\{1, \ldots, n\}$.
Initialize $V(1) = \pi(1)$ and the current value $E = w_{0,\pi(1)} + w_{\pi(1),0}$
for $i = 2, \ldots, n$ **do**
    select a position $pos \in \{1, \ldots i\}$ that minimize the energy value
    if $pos = 1$ **then**
        $E = (E - w_{0,V(1)}) + w_{0,\pi(i)} + w_{\pi(i),V(1)}$
    else if $pos = i$ **then**
        $E = (E - w_{V(i-1),0}) + w_{V(i-1),\pi(i)} + w_{\pi(i),0}$
    else
        $E = (E - w_{V(pos-1),V(pos)}) + w_{V(pos-1),\pi(i)} + w_{\pi(i),V(pos)}$
    end if
    Update $V(j+1) = V(j)$ for $j = pos, \ldots, i-1$ and $V(pos) = \pi(i)$
end for

In order to improve that heuristic we repeat the algorithm few times starting with different random orders. For ensuring the repeatability of the algorithm and the implementation on the user interface environment, we have implemented the Lehmer's algorithm for the Pseudo Random Number Generator [87].

3.4 Practical aspects

In order to improve the performance of the automatic function and to fit better the final application, we add two boolean information for each cooking process: the ice information and the dirty information.

3.4.1 Ice information

The ice information is a flag that identifies the cooking processes target for frozen foodstuff, for example frozen vegetables or frozen fish. The presence of frozen foodstuff will impact the energy consumption. When we are working with frozen foodstuff the shape of our objective function changes its slope if we are considering a warm-up phase (positive temperatures). Instead, the shape remains the same if we are during a cooling-down phase (negative temperatures).

In order to use this information, we have defined a corrective factor $k_{ice} > 0$ to increase the slope of the energy surface with respect to the temperature variable.

$$E_{pos}(\bar{T}, \Delta H) = k_{ice}(p_{1,1}\bar{T}\Delta H + p_{2,0}\bar{T}^2 + p_{1,0}\bar{T}) + p_{0,2}\Delta H^2 + p_{0,1}\Delta H + p_{0,0} \quad (3.3)$$

3.4.2 Dirty information

The dirty information is a boolean flag that identifies a cooking process with foodstuff that can soil the cavity of the oven, as grilled chicken or fish. Usually after these processes the chef has to clean up the oven in order to not contaminate the flavor of next cooking process. In alternative, all this cooking processes must be done at the end of the sequence, to reduce the impact on the other meals. The dirty information restricts the possible orders that we can arrange

The general request to postpone and move at the end of the sequence a set of cooking processes can be managed easily with the Karg-Thompson heuristic. Indeed, we just need to keep a pointer where the last "no-dirty" cooking process is. Each time we are processing a new cooking process, we will evaluate only the first half of the positions till the pointer, if the cooking process is not dirty, and the second half if the cooking process is dirty.

3.5 Example of Usage

The scheduling algorithm is meant for any food sector that is using the Cook & Chill process technique. One of this sectors is the catering sector. Let's consider a possible list of plates for a catering sector with a buffet menu. In Table 3.1 we list seven plates with the corresponding multistage cycle. We also add the ice information introduced in Section 3.4.1 with a boolean flag.

Using the polynomial function in Equation 3.2 we tune the coefficients with respect to a real Electrolux Professional oven. We obtain the polynomial surface in Figure 3.2. With these values we can apply the Karg-Thompson heuristic to obtain the optimal sequence. We can also search with an exhaustive search the worst sequence, i.e. the sequence with the maximum energy consumption. Since it would be unlikely that a chef would come up with the worst possible solution to begin with, we also decide to ask directly to a chef of the Electrolux Professional chef accademy to test also a realistic scheduling. We obtain the three sequence in Table 3.2.

In Figure 3.3, Figure 3.4 and Figure 3.5 we plot qualitative temperature target profiles for the best sequence, the custom sequence and the worst sequence, where we set each stage 1 to 5 minutes if it is a warm up and just 1 minute if it is a cooling down phase. We can immediately notice that in the worst sequence we continue to jump from a multistage with a low temperature to a multistage with high temperature, using lot of energy to heat up the oven. Then we loose everything by cooling down the oven and we need to heat up again for the next multistage. The custom sequence has a gentile constant warm up that reduces partially the waste of energy of the worst sequence but each recipe is not taking completely advantage on the heat accumulate in the previous step. Both the worst and the custom sequences end with

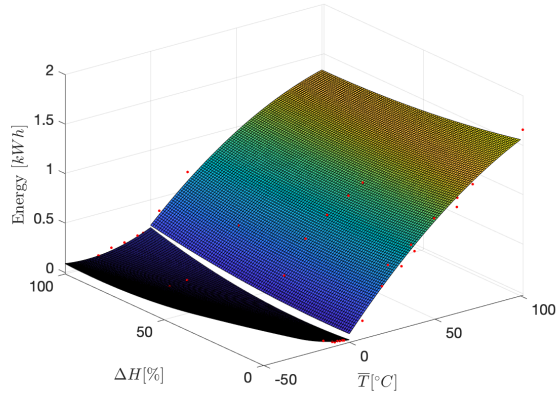


Figure 3.2: Polynomial function that approximates the energy consumption of a professional oven during the transient phases

Table 3.1: List of multistage cycles for a buffet menu. For each stage we list the temperature T, the humidity H and the duration t. If the duration is not specified, the stage ends when we reach the two conditions (T, H) inside the oven.

N.	Plate	Multistage parameters		Ice flag
		stage 1	stage 2	
1	Rice	$T = 100°C$ $H = 100\%$	$T = 100°C$ $H = 100\%$ $t = 13$min	0
2	Steam Vegetables	$T = 100°C$ $H = 100\%$	$T = 100°C$ $H = 100\%$ $t = 7$min	1
3	Cheesecake	$T = 140°C$ $H = 80\%$	$T = 120°C$ $H = 80\%$ $t = 15$min	0
4	Escalope	$T = 170°C$ $H = 70\%$	$T = 150°C$ $H = 70\%$ $t = 10$min	0
5	Pizza	$T = 200°C$ $H = 20\%$	$T = 180°C$ $H = 20\%$ $t = 11$min	0
6	Vegetable au gratin	$T = 230°C$ $H = 40\%$	$T = 210°C$ $H = 40\%$ $t = 18$min	0
7	Bacon	$T = 240°C$ $H = 10\%$	$T = 220°C$ $H = 10\%$ $t = 8$min	0

Table 3.2: Best sequence obtained with the Karg-Thompson heuristic, custom sequence made manually by a chef and worst sequence obtained with an exhaustive search. The numbers of the multistage systems refer to Table 3.1

Best	7	5	6	4	3	2	1
Custom	1	2	3	4	5	6	7
Worst	2	6	3	5	1	4	7

a high temperature and we need to cool down the oven, loosing all energy of this last multistage. In the best sequence instead we can notice a descendant shape for the temperature, where we heat the oven till the maximum, the most expensive step, and then we use this energy in the downhill.

The humidity profiles are reported in Figure 3.6, Figure 3.7 and Figure 3.8 using again the convention of 5 minutes for warm up stages and 1 minute when we have a cooling down stage. Again we can notice a jumping profile for the worst case and almost a monotone profile for the custom sequence and the best sequence. In the custom sequence, having selected a ascendent schedule for the temperature, we have the opposite trend in the temperature. The best sequence, instead, has an ascendant profile for the humidity and this is intuitively coherent with the effort to avoid the phenomenon of condensation that occurs when the oven is cold and we need to inject a high percentage of humidity.

Finally we notice that with the ice information the steam vegetables are performed at the end of the best sequence in order to use all the accumulated energy to faster defrost the vegetables. This phenomenon is not present on the reported qualitative figures but it has a huge impact in the real appliance.

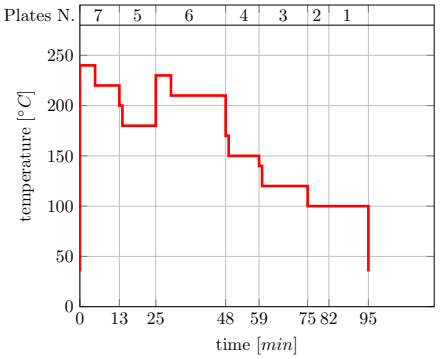


Figure 3.3: Temperature profile of the best sequence in Table 3.2

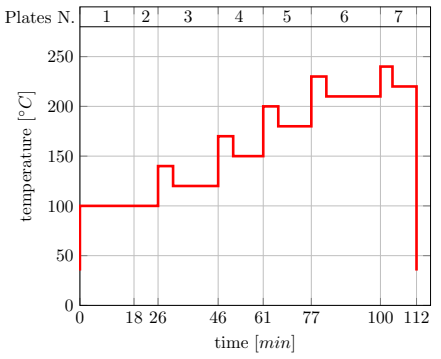


Figure 3.4: Temperature profile of the custom sequence in Table 3.2

To conclude the example we execute the three sequences inside a professional oven and we obtain -10% of energy saving for the best sequence with respect to the worst one and -6% with respect to the custom. The multistage cycles selected for the buffet menu have a variety of temperature and humidity stages and this highlights what we

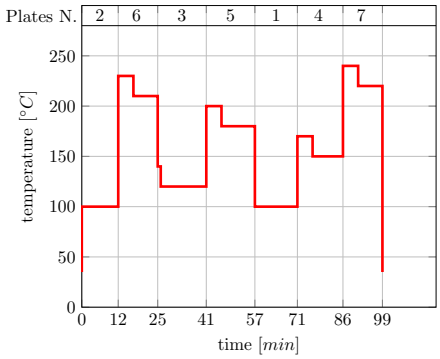


Figure 3.5: Temperature profile of the worst sequence in Table 3.2

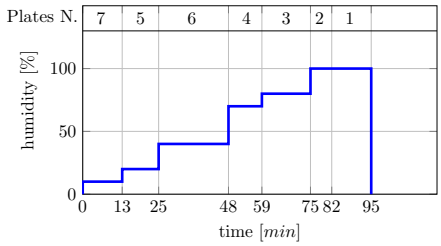


Figure 3.6: Humidity profile of the best sequence in Table 3.2

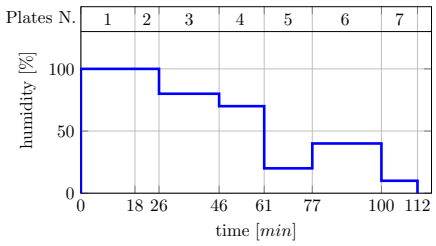


Figure 3.7: Humidity profile of the custom sequence in Table 3.2

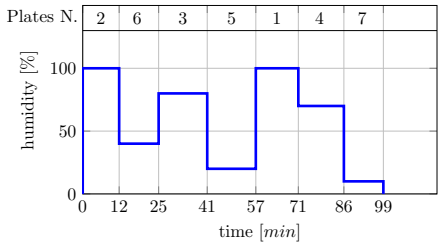


Figure 3.8: Humidity profile of the worst sequence in Table 3.2

can gain from the scheduling algorithm.

In general, this algorithm performs better with diversified multistage systems. On the contrary we will not save energy with a sequence of equal multistage systems because any possible order will have the same total energy. This is a characteristic of the scheduling problem and it depends on the definition of the energy function.

3.6 Conclusion and future works

We presented a scheduling algorithm for sorting a list of cooking recipes, i.e. a set of multistage systems, inside a professional oven. This new functionality works on the interaction between two consecutive multistage systems and suggest the best order that minimizes the total energy consumption of the machine. To reach this goal we solved a TSP considering the temperature and humidity of the first and last stage of each multistage system. We used an approximate model of the energy consumption of the oven to create the weight matrix and we finally solved the TSP instance with the Karg-Thompson heuristic.

The proposed solution is currently implemented in the Electrolux Professional ovens and it is increasing the use of smart solutions and assistant-making algorithms in the food service sector, particularly in Cook&Chill services.

This first step on the introduction of scheduling algorithm inside the food-service appliances opens an entire research field. We can think about the *central kitchens* where a set of machines is available in a common space for the production of different restaurants. Central kitchen operates strongly with the Cook&Chill process and the planning phase is crucial [108]. If we evolve the concept of central kitchen in a central kitchen hub, by connecting all the appliances and by controlling them remotely, the outright evolution of this study can be the scheduling on multiple appliances.

The TSP formulation can be easily extended to an hypothetical kitchen with multiple ovens of the same kind, it is enough to add a dummy state for each considered oven with a null weight on edges between two of these states. Since the current approximation of the energy consumption is a data-driven function that depends on the machine in witch we performed the tests, a different formulation is needed if we decide to extend further changing also size and typology of oven. In this situation could be interesting to change the energy function shape and parametrize the size or some characteristics of the considered machines.

4

Multi-objective black box optimization of a thawing process

Multistage heat transfer processes are a family of physical processes where the behaviour of the thermal system is dependent on a specific combination of parameters. How to select the most suitable combination, according to a given objective function or to a specific requirement, is highly influenced by the complexity of the problem, the nature of the parameters (number and range) and their role in the physical process.

We want to focus on a special multistage heating process, the *thawing* process, that is a food service process where food is subject to a heat transfer operation with the external environment, in a quantity sufficient to just pass the phase transition of the water present in the food matrix from solid to liquid. In professional appliances, commonly used in professional kitchens, chains or canteens, the thawing process is executed under controlled conditions, for example in terms of airflow temperature and airflow velocity. The temperature is typically kept in the range $4°C$ - $15°C$ to prevent the bacterial growth [37]. Professional appliances allow to program a multistage thawing cycle. Each stage is characterized by a set of parameters, as the air temperature and the duration of the stages.

The selection of the parameters has to balance the productivity and quality of the process. Indeed thawing of raw fish and meat can affect the physical, chemical and microbiological quality of the product, and thus remains a major concern for processors and consumers [3, 37]. The bacteria that cause food poisoning are found on the surfaces of many foods and, in this process, they start growing due to the continuous arise of the temperature of these areas. There are different evolution models that take into account the bacterial growth depending on the temperature and the duration, mostly variants of the logistic evolution model or piecewise exponential growth curves obtained with empirical methods [88]. The temperature difference between the food and the surrounding medium is important to prevent the development of microbial flora and to reduce the impact of enzymatic activity. In our case we choose to consider the most general and the strictest constraint that we can impose. The same construction can be extended to more complex models. In summary, we want to keep

the maximum temperature $T_{\max}$ of the product below a T_{safe} temperature of $7°C$ following the codex alimentarius for fish fillet [59].

The optimization problem described is quite general. We have a multistage system and we want to minimize the total duration of the process and the temperature unevenness within the food having an end condition—in the thawing case is a minimum threshold that we have to reach, i.e. the minimum temperature $T_{\min}$ inside the product has to reach $0°C$—and satisfying a constraint that in this case is a maximum threshold that we do not want to overpass ($T_{\max} \leq 7°C$).

The heating process is described by the computationally efficient model in [103]. This model is able to describe the multi-phase transient field inside the food object, by means of a finite difference numerical scheme [36]. The only operation that we can perform with this model is the evaluation, namely we have a procedure to evaluate the model but we do not have an analytic expression for it. Such a function is called *black box function.*

Definition 4.0.1. *A* black-box optimization problem *is an optimization problem in which the objective function is a black box function.*

Following [81] and [90] we can broadly divide all the algorithms used to solve a black-box optimization problem into two classes:

1. Direct method
2. Surrogate-based methods.

We refer to *direct methods* when the algorithm determines search directions by evaluating the objective function directly, whereas *surrogate-base algorithms* create and use a surrogate function to guide the search process, i.e. an analytic function that approximates the complex system.

Direct methods are advisable when we have a computationally efficient model. In this scenario the main problem is to deal with black box constraints that express a property on the trajectory. The difficulties arise because potential solutions, that violate black box constraints, provide no information beyond their infeasibility. A possible solution generally used is to relax the constraint and to treat it as an additional objective function. On the other hand, the use of a surrogate-based method allows to handle the constraint directly and it gives us the possibility to exploit the graph structure behind the multistage system.

We decide to exploit both the possibilities. In Chapter 4.1 we investigate a direct method and the Genetic Algorithms (GA) are a reasonable direct method to analyze a multi-objective optimization problem. A surrogate-based method is analyzed in Chapter 4.2, where we consider the multistage system as a single hybrid automaton.

4.1 Genetic algorithm approach

We conduct a multi-objective optimization analysis for a jet impingement thawing process which is described by means of a computationally efficient model. The availability of computationally efficient models allows to speed up the research of the right

configuration, thanks to the possibility to analyze a huge amount of test cases. The physical behavior of the thawing model and the nature of the input parameters may introduce multimodal trends of the objective functions. In such cases only a robust optimization algorithm can ensure a good convergence without being trapped at a local minimum. Moreover we are interested also in obtaining a set of uniform solutions on the Pareto front, in order to select some promising candidates and investigate the right compromise between the objective functions. In literature, evolutionary algorithms are quite recognized as state of the art, able to deal with the challenges described above [101, 26, 40, 84, 75, 2].

[84] present a literature review about the most used optimization algorithms for building energy simulations, showing a significant use of GA among 200 building optimization studies retrieved by SciVerse Scopus of Elsevier. They remark that the popularity of GA is mainly due to its capability and robustness in dealing with problems of different nature, i.e. optimizing functions with continuous or discrete variables, accelerating the search with parallel simulations, working with populations, robust handling of discontinuous, multimodal and highly constrained objective functions. Also [40] confirms the trend about the use of GA algorithm in optimization studies for building energy simulations, showing firstly a predominance of evolutionary algorithms and secondly the extensive use of GA for more than half the analyzed articles. Moreover, based also on the work of [91], he confirms that evolutionary algorithms represent the best compromise to handle an optimization study for a complex black box model.

The extensive use of GA and the widespread references about its robustness and reliability, drive our choice in the selection of the optimization algorithm. Moreover having an algorithm compatible with the parallel computation is extremely important to save time and money during the product development process in a company. MATLAB®offers the possibility to parallelize the GA algorithm among the individuals of each generation in an effortless way, by simply enabling the parallel option of the algorithm [76].

As other meta-heuristic techniques, GA requires the setup of the internal parameters, whose values are dependent on the particular problem, as remarked by [98]. This is true especially for black box functions, since we have poor information about the function itself. As successfully shown by [75], [99] and [8], the setup configuration of the GA can be tuned by using the *Taguchi design method*. Their works are focused on single-objective optimization problems. We studied the importance of tuning the parameters of a GA by using the Taguchi design method also for multi-objective problems, showing firstly its efficiency for some benchmark cases and then implementing the method for the thawing case. In order to compare different Pareto fronts, we evaluate the performance of each configuration using the *hypervolume indicator* introduced in Chapter 1.2.1. We start with the description of the model and then we describe the procedure to setup the parameters.

4.1.1 Jet impingement thawing model

In the last few years the jet impingement technology in food service is highly under research because the use of forced air-streams seems to have advantages compared

to other traditional methods [9, 93, 79]. The impingement systems exploit jets of fluid, at high velocities, ideally impinging perpendicularly the surface of products [97]. Impingement reduces the thermal boundary layer thickness surrounding the products surface, increasing the convective heat transfer coefficient and allowing faster heat transfer processes. Jet impingement is potentially advantageous for increasing the thawing rate without increasing the air temperature, when microbial growth becomes a concern. In [103] a model for jet impingement thawing appliances is proposed. The model combines in cascade a Computational Fluid Dynamic model and a Finite-Difference Heat-Conduction model and it is validated on a Tylose brick, a common food analog. This model describes the thermal evolution of a brick of dimension $16 \times 8 \times 5$ cm^3. The brick is divided in about 1000 blocks and the model describes the temperature evolution $T(x, y, z, t)$ of each block with the *enthalpy method*. We can treat the selected thawing model as a function

$$\mathcal{T} : \mathbb{R}^n \to \mathbb{R}^3 \tag{4.1}$$

where n is the number of input needed to describe a particular thawing cycle and the three output are the three objective functions that we want to optimize. If we setup a multistage cycle with at most m stages, we have the following input:

1. Re: Reynolds number that expresses the air speed. $Re \in [12000, 26000]$.
2. HD : dimensionless distance from the nozzle to the surface of the food (H distance, D nozzle diameter). $HD \in [2, 6]$.
3. $T_{set} = (T_1, \ldots, T_m)$: vector of dimension m with all the air temperatures expressed in Celsius. $T_{set} \in [4, 15]^m$.
4. $time_{set} = (t_1, \ldots, t_{m-1})$: vector of dimension $m - 1$ with all the durations per each setpoint expressed in seconds. $time_{set} \in [0, 3600]^{m-1}$. The last setpoint does not have a duration but we keep it till the end of the process, so till the minimum temperature $T_{\min}$ within the food item reaches $0°C$.

The first two input are peculiar of the jet impingement technology, the other input instead are common to every multistage thawing process.

The three output of the model that we evaluate for each multistage thawing cycle are the following:

1. Σ_t: total time necessary to complete the process, expressed in hours.
2. UI: Uniformity Indicator that measures the temperature uniformity within the food item during the thawing process.

$$UI = \frac{1}{\Sigma_t} \int_t \frac{\int_V \mid T - \mu(T)) \mid dV}{V(T_{tg} - T_0)} dt \tag{4.2}$$

where T means $T(x, y, z, t)$ and it is the temperature of the block in position (x, y, z) at time t, $\mu(T)$ is the average temperature of the food item of volume V at time t. T_0 is the food item initial temperature and T_{tg} is the target setpoint

temperature for $T_{\min}$ that is $0°C$. UI is dimensionless and it ranges between 0 and 1, where 0 means that the temperatures inside the food are completely uniform.

3. $T_{\max}$: maximum temperature within the food item reached during the process, expressed in Celsius.

$T_{\max}$ is the indicator that we have chosen to keep the bacterial growth under control.

Through the user interface of the professional appliance usually we can setup a multistage cycle with a finite number of stages. Realistically, we can decide to consider at most 5 stages. We can assume that we have always a vector of 11 input. In conclusion we treat the thawing model as the function

$$\begin{aligned} \mathcal{T} : \mathbb{R}^{11} &\to \mathbb{R}^3 \\ x &\to (\Sigma_t, UI, T_{\max}) \end{aligned} \tag{4.3}$$

where $x = (Re, HD, T_1, \ldots, T_5, t_1, \ldots, t_4)$ and we denote with $\mathcal{T}(x) = (\Sigma_t(x), UI(x), T_{\max}(x))$.

The following example will clarify better the meaning of the input and output variables. We will consider a case A with the following input parameters: $Re = 12000$, $HD = 6$ and a profile of the setpoint temperature of the airflow considering $T_1 = T_2 = T_3 = T_4 = 5°C$ and $T_5 = 4°C$ and the setpoint time $t_1 = t_2 = t_3 = t_4 = 900s$. Then we will consider a case B with the following setup: $Re = 18000$, $HD = 2$ and a profile of the setpoint temperature of the airflow considering $T_1 = T_2 = T_3 = T_4 = 15°C$ and $T_5 = 7°C$ and the setpoint time $t_1 = t_2 = t_3 = t_4 = 720s$. The test case A is meant to provide a lower thawing rate (slow air speed and low air temperature), but a more uniform temperature distribution within the food, on the contrary the case B will provide a very fast thawing rate (high speed and high temperature), but with a very uneven temperature distribution and risk of hot-spots. Figure 4.1 clearly shows the difference between the two cases in terms of thawing rate, by looking at the profile of the minimum temperatures within the food. The case A takes $\Sigma_t = 3.82h$ to end the thawing process versus $\Sigma_t = 1.43h$ of the case B. Figures 4.2 and figure 4.3 show the temperature contour map at a mid-section of the food item at the end of the thawing process. It is clear that the case A presents a more uniform temperature distribution $UI = 0.10$ and the maximum temperature is $T_{\max} = 4.1°C$, while the case B presents a very uneven temperature distribution $UI = 0.31$ and its maximum temperature reaches $T_{\max} = 13.3°C$, clearly above the safety limit.

Given the model $\mathcal{T}$ for the thawing process we want to find the optimal thawing cycle, so the optimal input x, that minimizes $\Sigma_t(x)$ and $UI(x)$ keeping $T_{\max}(x)$ below T_{safe}.

We write the associated multi-objective optimization problem $\mathscr{P}$ for the thawing case.

$$\begin{aligned} \mathscr{P} : \quad & \min_x \ (\Sigma_t(x), UI(x)) \\ & s.t. \ \ T_{\max}(x) \leq T_{safe} \end{aligned} \tag{4.4}$$

As already pointed out, the only operation that we can perform for the function $\mathcal{T}$ is the evaluation. Thus we need to relax the constraint and to treat it as an additional

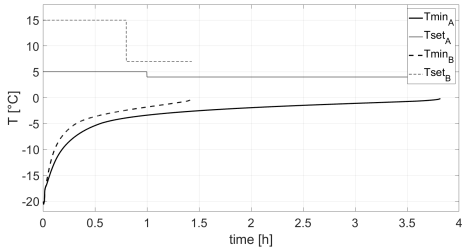


Figure 4.1: Case A and case B: set point temperatures of the air and minimum temperatures within the food item.

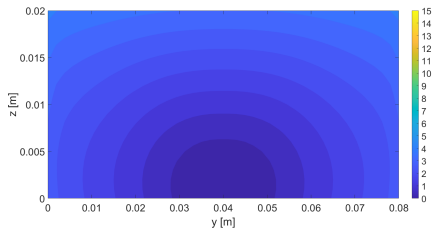


Figure 4.2: Case A: end-process temperature contour map in Celsius at the mid-section of the food item.

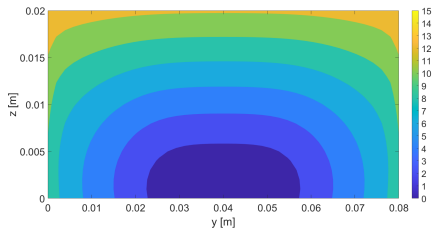


Figure 4.3: Case B: end-process temperature contour map in Celsius at the mid-section of the food item.

objective function. In this case we want to keep $T_{\max}$ under a certain level, so we ask to minimize the value of this third indicator. In conclusion we are going to solve the following problem

$$\mathscr{P}_{\mathcal{T}}: \quad \min_{x} \left(\Sigma_t(x), UI(x), T_{\max}(x)\right) \tag{4.5}$$

The problem $\mathscr{P}_{\mathcal{T}}$ has three functions to be minimized, so the solutions are all the points on a three dimensional Pareto front. GA is a reasonable direct method to obtain a set of points that approximately lies on the Pareto front. Moreover, using the software MATLAB®, we have the possibility to implement a multi-objective optimization problem with GA using the function `gamultiobj` [76].

4.1.2 Parameters setup for a genetic algorithm

Genetic algorithms are optimization algorithms that mimic the biological process of reproduction and natural selection. The basic components of a GA are:

1. *population*: batch of solutions that we are currently evaluating. Each solution $x = (x_1, \ldots, x_n)$ of the problem is an individual of the population and its components x_i are called chromosomes.

2. *selection*: how we select two individuals to be recombined.

3. *crossover*: it is the operation to recombine two individuals to obtain the new one.

4. *mutation*: it is the operation to change randomly a single chromosome of an individual.

We have also to choose which is the starting population and how huge it is. At each run of the GA we use the rules of selection, crossover and mutation to create the new generation of individuals.

The function `gamultiobj` is highly customizable. We have some default functions for each component of the GA but we can also implement our own functions. In Table 4.1 we recap the 7 parameters we have decided to tune and the options considered for each parameter.

We have 7 parameters or factors, 2 of them have 4 options or levels and the others have just 2 levels. All the possible combinations are $4^2 \cdot 2^5 = 512$. In most of the research studies these settings are based on either default values, or reference values from the literature or trial and error approaches. When reference values from literature are not available, a tuning phase of the default settings is advisable. In the present work the Taguchi design method is used for this purpose. Taguchi method is a statistical method largely used in engineering contexts to reduce the number of tests in a robust way. In particular the Taguchi method allows also to work with factors that have different number of levels. The details of the 16 tests that we perform are summarized in Table 4.2.

In order to evaluate the sixteen tests, we need to use a performance indicator able to compare two different Pareto fronts. We choose to compare two Pareto fronts using the *hypervolume indicator* I_H defined in Chapter 1.2.1. This indicator measures the

Table 4.1: Matlab genetic algorithm parameters. Functions with * are custom functions that we implement. Directional crossover function is explained in [46]

	Parameter	No.	Option
A	Crossover function	1	Single Point
		2	Multi Point
		3	Heuristic
		4	Directional*
B	Crossover Fraction	1	0.7
		2	0.8
		3	0.9
		4	0.95
C	Mutation function	1	Adaptive
		2	Random*
D	Creation function	1	Linear
		2	Random*
E	Migration Fraction	1	0.0
		2	0.2
F	Pareto Fraction	1	0.10
		2	0.35
G	Population size	1	100
		2	200

Table 4.2: Taguchi design obtained with commercial software [78]

Test	Parameter of GA						
	A	B	C	D	E	F	G
1	1	1	1	1	1	1	1
2	1	2	1	1	1	2	2
3	1	3	2	2	2	1	1
4	1	4	2	2	2	2	2
5	2	1	1	2	2	1	2
6	2	2	1	2	2	2	1
7	2	3	2	1	1	1	2
8	2	4	2	1	1	2	1
9	3	1	2	1	2	2	1
10	3	2	2	1	2	1	2
11	3	3	1	2	1	2	1
12	3	4	1	2	1	1	2
13	4	1	2	2	1	2	2
14	4	2	2	2	1	1	1
15	4	3	1	1	2	2	2
16	4	4	1	1	2	1	1

hypervolume of the portion of the objective space that is dominated by the set of points on the Pareto front. In order to measure this quantity, the objective space must be bounded or, alternatively, we should use a reference point that is at least weakly dominated by all points.

According to the setup decided for the multistage thawing cycle, the computationally efficient model [103] is able to describe the multi-phase transient field inside the food object, by means of a finite difference numerical scheme [36]. In our case we have an upper bound and a lower bound for all the three objective functions. The maximum temperature is dependent on the air temperature so it can not exceed $15°C$ and it can not be lower than the target of $0°C$, thus $T_{\max} \in [0, 15]$.

We can prove easily that UI is in the interval $[0, 0.5]$. The indicator is non-negative by definition and it is 0 only if each temperature T inside each volume V has the same temperature at each instant t. The upperbound instead is a conservative estimation based on the fact that the extreme condition for the value $|T - \mu(T)|$ is obtained when $\mu(T) = \frac{(T_{tf} - T_0)}{2}$ and half of the volumes have $T = T_0$ and the other half have $T = T_{tg}$. But this can be reached only in some time instances, not in all, because of the end condition imposed to the system.

As regard the total time we can estimate the range knowing the best and the worst thawing cycles with respect to Σ_t. For the best thawing cycle C_{best} we select the maximum air speed, the maximum air temperature and the smallest distance from the jets to the food surface. Instead for the worst case C_{worst} we select the minimum air speed, the minimum air temperature and the highest distance from the jets to the food surface.

$$C_{best} = [26000, 2, 15, \ldots, 15, 3600, \ldots, 3600]$$
$$C_{worst} = [12000, 6, 4, \ldots, 4, 3600, \ldots, 3600]$$

We decide to normalize the three objective functions in $[0, 1]$. With this new domain, the Pareto front is in $[0, 1]^3$ and the bounding point is $(1, 1, 1)$ so also the indicator I_H is normalized in $[0, 1]$.

4.1.3 Results for benchmark problems

We run the GA with the 16 different setups and with a small total number of generations fixed to 300, we calculate I_H on the resulting Pareto front and we repeat each setup 3 times in order to take the average value and in order to increase the consistency of the results.

The results for Taguchi design are analyzed according to the signal-to-noise (S/N) ratio. It is a measure of variation and it guarantees the robustness of the design [77]. I_H is an indicator that we want to maximize, thus the S/N is defined as

$$(S/N)_i = -10 \log_{10} \left(\frac{\sum_{k=1}^{3} \left(\frac{1}{Y_{i,k}^2} \right)}{3} \right) \tag{4.6}$$

where $Y_{i,k}$ is the value of I_H for each test i and repetition k and we sum $\frac{1}{Y_{i,k}^2}$ over the three repetitions.

Before applying this procedure to the thawing problem, we first consider the same strategy with six benchmark problems, to empirically evaluate the validity of the method.

We consider 6 benchmark problems and we compare the results obtained with the proposed procedure and the results obtained with the default GA of MATLAB®. With this comparison we want to stress the fact that a GA must be customized for each problem, if we want to be confident of the result, and Taguchi with I_H is an efficient way to proceed even with multi-objective functions.

We select benchmark problems DTLZ1-6 [31] for the ability to define both decision variables and objective functions to any dimension. Indeed we define these problems with exactly 11 variables and 3 objective functions, like the thawing problem. These problems are widely used to test GA performance because each problem introduces a different challenge as the convergence to the ideal Pareto front or the diversity of the final solutions or the ability to keep a widely distributed set of solutions for the last generation with respect of the objective functions.

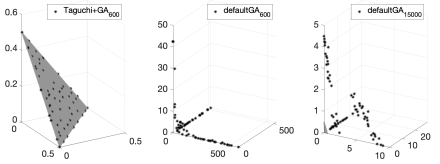


Figure 4.4: DTLZ1: Comparison between the obtained front (black) and the ideal Pareto front (gray).

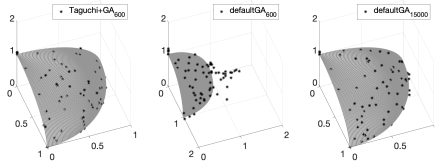


Figure 4.5: DTLZ2: Comparison between the obtained front (black) and the ideal Pareto front (gray).

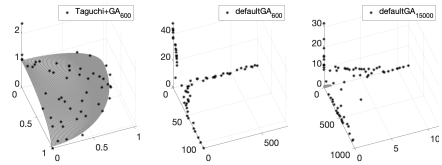


Figure 4.6: DTLZ3: Comparison between the obtained front (black) and the ideal Pareto front (gray).

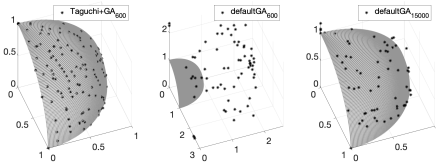


Figure 4.7: DTLZ4: Comparison between the obtained front (black) and the ideal Pareto front (gray).

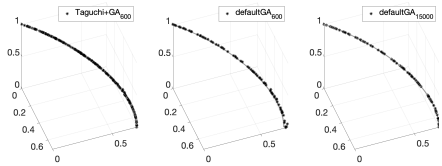


Figure 4.8: DTLZ5: Comparison between the obtained front (black) and the ideal Pareto front (gray).

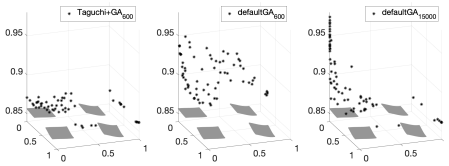


Figure 4.9: DTLZ6: Comparison between the obtained front (black) and the ideal Pareto front (gray).

After selecting the best setup following the described procedure, we run the GA with the best setup for 600 generations. We compare the result for the best setup with the result of the default setup run for 600 generations. We also compare the default setup with our method by considering almost the same of computational time, running the default setup for a number of generations equal to the sum of all the generations of the Taguchi design adding also the number of generations used for the run with the best setup (i.e. $300 \cdot 16 \cdot 3 + 600$). The knowledge of the ideal Pareto front permits us to better compare the results. First of all we compute I_H for each resulting front and we consider the difference with the hypervolume of the ideal Pareto front (I_H^i). Then we also consider two additional indicators following [120, 28] and [58]:

- the generational distance indicator (GD) to evaluate the coverage [110]

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{4.7}$$

  where n is the number of obtained solutions, d_i is the Euclidean distance in objective space between the ith solution and the nearest solution on the ideal Pareto front.

- the uniformity and diversity indicator Δ introduced by one of the inventors of the benchmark problems [30]

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{n-1} |d_i - \bar{d}|}{d_f + d_l + (n-1)\bar{d}} \tag{4.8}$$

  where n is the number of obtained solutions, d_f and d_l are the Euclidean distance between the extreme solutions of the ideal Pareto front and the obtained solutions, d_i is the minimum distance from one solution to the next one (in the objective space) and $\bar{d}$ is the average of all distance d_i.

All the three indicators must be minimized. The results are reported in Table 4.3 and we highlight in bold the best result for each indicator. We also plot the Pareto front in the three different scenarios. Looking at the figures, we can immediately notice that, in DTLZ1 problem (Figure 4.4) and in DTLZ3 problem (Figure 4.6), the default setup is not able to converge even in the huge scenario of 15000 generations. Instead, with the Taguchi tuning, in just 600 generations we are already near the ideal Pareto front in all the 6 benchmarks.

4.1.4 Results for the thawing model

Following the same method for the thawing case, we run the GA with the 16 different setups, 3 times each, and with a maximum number of generations fixed to 300. We perform each run on a workstation with Intel® Xenon® E-2176M CPU at 2.7GHz, 6 cores, 32 GB RAM and using MATLAB®R2019b with Global Optimization Toolbox and Parallel Computing Toolbox. Each run takes in average 2014 seconds and at each call, the black box model is evaluated in 0.142 seconds in average. The calculated I_H is reported in Table 4.4.

Table 4.3: Comparison between the setup obtained after a tuning with Taguchi (Taguchi+GA_{600}), the default setup of MATLAB®with the same number of generation of our final run (GA_{600}) and the default setup with a same number of generations of our total method (GA_{15000}).

		Taguchi+GA_{600}	defaultGA_{600}	defaultGA_{15000}
DTLZ1	$I_H^i - I_H$	$\mathbf{1.1206e-01}$	$4.7518e+04$	$5.9996e+01$
	GD	$\mathbf{3.0491e-04}$	$3.2624e+01$	$9.2207e-01$
	Δ	$\mathbf{1.9527e-05}$	$2.9986e-04$	$3.0085e-04$
DTLZ2	$I_H^i - I_H$	$\mathbf{1.0344e-01}$	$3.4478e-01$	$1.2049e-01$
	GD	$\mathbf{2.4413e-03}$	$2.5246e-02$	$3.1507e-03$
	Δ	$4.9191e-07$	$2.2968e-04$	$\mathbf{2.2900e-10}$
DTLZ3	$I_H^i - I_H$	$\mathbf{8.8061e-01}$	$5.4195e+04$	$2.7230e+03$
	GD	$\mathbf{2.1358e-02}$	$3.3370e+01$	$3.3920e+01$
	Δ	$\mathbf{9.9503e-05}$	$3.0542e-04$	$3.1098e-04$
DTLZ4	$I_H^i - I_H$	$\mathbf{7.5104e-02}$	$9.4333e+00$	$1.4717e-01$
	GD	$\mathbf{5.5542e-04}$	$1.8808e-01$	$4.7269e-03$
	Δ	$9.9358e-06$	$2.9049e-04$	$\mathbf{5.0017e-08}$
DTLZ5	$I_H^i - I_H$	$\mathbf{5.2905e+00}$	$5.5266e+00$	$5.3719e+00$
	GD	$\mathbf{2.6292e-04}$	$1.3243e-03$	$3.3226e-04$
	Δ	$2.5592e-05$	$8.9160e-03$	$\mathbf{6.4881e-10}$
DTLZ6	$I_H^i - I_H$	$\mathbf{1.6742e-02}$	$2.8244e-02$	$1.7003e-01$
	GD	$1.7398e-02$	$\mathbf{1.1568e-02}$	$1.3277e-02$
	Δ	$\mathbf{3.6501e-04}$	$5.3343e-04$	$3.8945e-04$

The hypervolume indicator has to be maximized and we can immediately notice that some combinations are less promising than others: for example all the three values obtained for the test 1 are smaller than the I_H for each run of the test 4. However, a definitive choice about the best parameters configuration can be made only studying the cross-correlation effect by a statistical indicator.

Table 4.4: Results obtained from each test and each repetition

Test	I_H value		
	Run 1	Run 2	Run 3
1	0.35516	0.30559	0.36710
2	0.50232	0.50346	0.49503
3	0.45930	0.41904	0.44885
4	0.50056	0.50744	0.50694
5	0.46632	0.47957	0.47356
6	0.48476	0.47415	0.47881
7	0.45463	0.46700	0.44785
8	0.45942	0.46615	0.46931
9	0.48976	0.48362	0.48349
10	0.47303	0.48696	0.48417
11	0.46851	0.49273	0.48594
12	0.44424	0.45557	0.46504
13	0.49365	0.49698	0.50448
14	0.40960	0.29412	0.42170
15	0.50690	0.50792	0.50928
16	0.47190	0.46685	0.46988

Table 4.5: Selection of the MATLAB®Genetic Algorithm parameters.

Parameter	Type
Crossover function	Heuristic
Crossover Fraction	0.95
Mutation function	Adaptive
Creation function	Linear
Migration Fraction	0.2
Pareto Fraction	0.35
Population size	200

Following Equation (4.6), we compute the signal-to-noise (S/N) ratio for each option of the parameters of the GA, so for each level and each factor of the Taguchi design. This indicator again has to be maximized. The main effects plot is shown in Figure 4.10 where we calculate the average of S/N for each level and each factor.

For each factor $A,\ldots,G$ in Figure 4.10 we have to select the level with the maximum S/N. Starting from the factors that have the most evident impact, we should consider the migration (factor E) by setting a value greater than 0 (in this case we

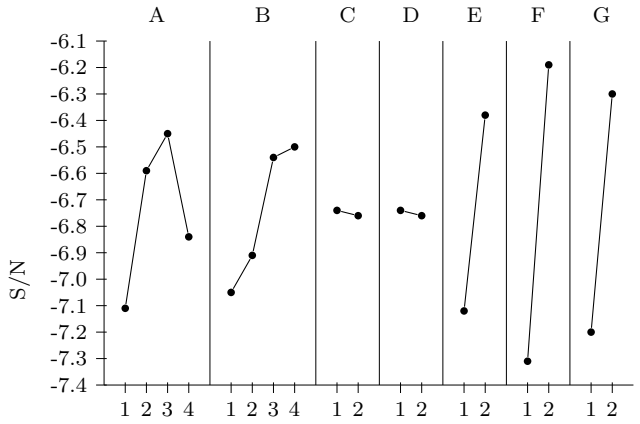


Figure 4.10: Main effects plot with the average S/N values of the I_H for each level and each factor.

considered 0.20), the Pareto fraction (factor F) should be set at 0.35 (level 2) instead of 0.1 and we should consider a population (factor G) of 200 individuals (level 2) instead of 100. It is also clear that we should not use the single-point function for crossover (factor A, level 1) and we should use a high crossover fraction (factor B). The effect of the other factors or levels is less evident but we select in any case the level with the highest value. We run the GA with the parameters in Table 4.5 with a maximum number of generations fixed at 600.

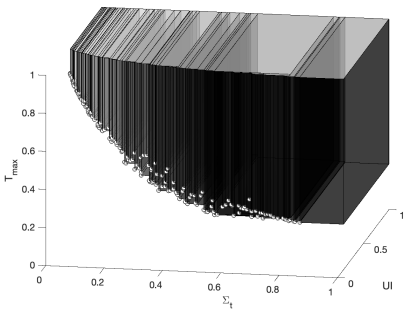


Figure 4.11: Slices obtained to calculate the I_H for the points on the optimal Pareto front

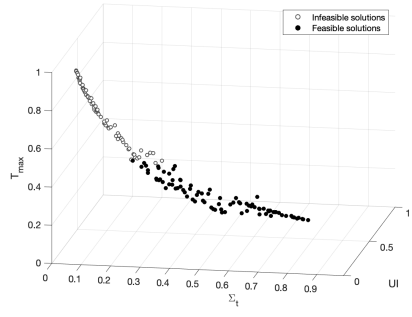


Figure 4.12: Individuals on the optimal Pareto front: black points are feasible solutions for the problem $\mathscr{P}_{\mathcal{T}}$

The resulting Pareto front presents a hypervolume $I_H = 0.51$, visible in Figure 4.11. All these individuals are solutions of the optimization problem $\mathscr{P}_{\mathcal{T}}$ but not all of them are feasible for the original problem defined in Equation (4.4). If we restore the constraint condition $T_{\max} \leq 7$ we have to prune half of the solutions and we obtain only the individuals with black markers in Figure 4.12. Moreover we can observe

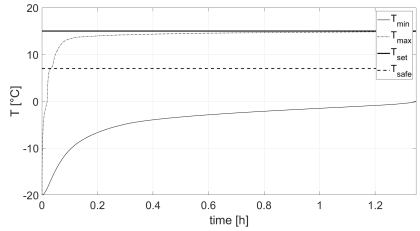


Figure 4.13: Minimum and maximum temperature evolution within the food item for the cycle C_{best}

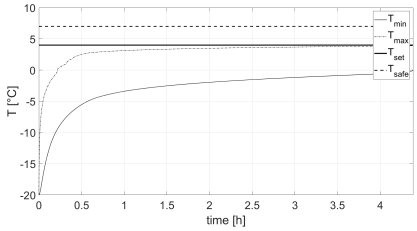


Figure 4.14: Minimum and maximum temperature evolution within the food item for the cycle C_{worst}

that Σ_t seems inversely correlated with $T_{\max}$. Indeed we obtain as extreme points the two thawing cycles C_{best} and C_{worst}. The presence of these two cycles is extremely important, because, from the theory of the heat transfer process within the food item, they are the theoretical extreme cycles that minimize Σ_t (C_{best}) and minimize $T_{\max}$ (C_{worst}). Consequently they theoretically belong to the ideal Pareto front and this increases the confidence of the goodness of the obtained Pareto front. In Figure 4.13 and Figure 4.14 the evolution of the minimum and maximum temperatures within the food item for C_{best} and C_{worst} are plotted over time. Again, it can be observed that the maximum temperature of C_{worst} does not satisfy the constraint defined in Equation 4.4.

Another peculiarity of the final generation is the redundancy of some individual. In particular, when we consider a single stage process that lasts less then 4 hours, some of the final temperatures in T_{set} are assigned randomly without having an impact on the three output. So we can have two input with the same triple output. This is mainly due to the fact that we are obliged to consider an individual with a fixed number of chromosome. Nonetheless the upper bound of 1 hour for the values of $time_{set}$ has reduced the redundancy problem.

4.2 Hybrid automaton approach

In the previous sections we have seen a possible strategy to solve a multi-objective optimization problem when we have a black box model, in the end we obtain a set of points on the Pareto front and we prune with respect to the constraint. Here we want to discuss a possible alternative by reformulating the problem as a reachability problem for a hybrid system.

Lifting up our view from the single application, here we want to focus on the problem of parametrically designing what we can call *multistage systems*, that are systems characterized by n stages, where each stage has a precise—albeit parametric—evolution law. We give multistage systems as a single hybrid automaton [6], whose states are grouped in *layers* and states are characterized by a continuous (parametric) evolution law. Each layer qualitatively represents a stage of the entire process and the discrete transition represents the switching instant from one stage to the next

one. The underlying structure of the automaton will be that of a *Directed Acyclic Graph* (DAG) and global properties of the process will be optimized with respect to full paths in such a DAG.

A multistage system is, ultimately, a parametrized hybrid system. In the literature we can find two main classes of problems analyzed on these systems: *parameter synthesis* and *optimal control design*. Parameter synthesis is simply the problem of choosing a set of parameters in order to ensure a given property. In [38, 34, 25] (among many others) we can find possible approaches to solve this problem. Optimal control design is the problem of choosing the best *trajectory* that satisfies a given property, where with trajectory we mean a possible (full) evolution of the system. Also here we have many examples of possible approaches, such as [70, 16, 53]. The thawing problem is a combination of the two: we want to select a setting of parameters suitable to obtain the best trajectory that satisfies a given property.

In the literature this problem goes under the name of *optimal parameter synthesis* (or *safe reachability problem*) and is tackled specifically in [7, 17]. In [17] the authors perform optimal parameter synthesis but they are working with *transient* systems and not with multistage systems as in our case. In a transient system you just have the set of states (finite or infinite) and a transition relation expressed with a formula, that in [17] is a linear arithmetic formula. They use model-checking to obtain the set of all admissible solutions and they illustrate different methods to explore the solution-space in order to find the best solution. In [7], instead, are addressing a scheduling problem for a reducing peak power demand of the heating system of multiple zones for a commercial building [83]. In this problem a solution is an *un*ordered list of parameters with the associated duration. The authors are not interested in the specific stage for which we select a given parameter, for this reason they can reduce the optimal parameter synthesis to a *linear* optimization. Finally in [52] they address a different but related problem, a scheduling problem, with non-linear cost functions. They reformulate it over modified priced timed automata with non linear cost function in each location. Similarly to what we are going to propose, they found the solution with a reachability analysis.

Here, we propose a procedure to solve optimal parameter synthesis problem working directly on the hybrid automaton. We work on the structure of the automaton in order to reduce the optimal parameter synthesis on a multistage system to the optimal control design on an automaton with an augmented number of locations—that we will call *multistage automaton*. Doing this we address the optimization problem as a reachability problem and we use a backtrack strategy [5] to obtain a feasible solution in the linear case.

The target is to solve the optimal parameter synthesis for a multistage cooking program. The procedure we describe is very general and can be applied to every multistage system. For example, we can create a continuous "cutting-stock" problem thinking of a painting product-line of a metal tube factory, where we want to maximize the overall length of tubes produced, while keeping the paint consumption for each primary color under a supply limit threshold. As another example, we can apply our methodology to find an ordered solution for the peak power demand problem described in [83].

Let us consider the thawing problem defined in Chapter 4.1.1. To simplify the

discussion we decide to not consider the air flow speed and the distance from the nozzle, we fix these two input to $Re = 26000$ and $HD = 2$, and we want to minimize just the total time. In general we can set m different *stages*, namely m different temperatures of the cavity in the form of a vector $T_{set} \in [4, 16]^m$ with the temperatures expressed in Celsius unit. In addition we require how long we want to keep these temperatures, so we fix another vector $time_{set}$ with $m - 1$ duration expressed in seconds.

The optimization problem described is quite general. We have a multistage system with $m(m-1)$ variables and we want to minimize the total time having an end condition—in the thawing case is a minimum threshold that we have to reach ($T_{\min} = 0°C$)—and satisfying a constraint that in this case is a maximum threshold that we do not want to overpass ($T_{\max} \leq 7°C$).

Example 4.2.1 (Linear multistage system)**.** *We introduce the following simplified (linear) version of the thawing cycle. Suppose we have only* 3 *stages and suppose we deal with two variables x_1, x_2 only, evolving linearly. x_1 is the minimum temperature and x_2 is the maximum temperature. Suppose $T_{set} \in \{4, 8, 12, 16\}^3$, so we have to select one of these temperature for each of the 3 stages. If we are in the stage i we select the temperature $T_{set}(i)$ and we have:*

$$x_1(t) = x_1(0) + \frac{(T_{set}(i) + 3.2)}{4800} \cdot t \qquad x_2(t) = x_2(0) + \frac{T_{set}(i)}{2200} t; \tag{4.9}$$

The starting values $x_1(0)$ and $x_2(0)$ for the first stage are $-18°C$ (typically the freezer temperature). The starting value in the second and third stage are the last value of the previous stage. In this example choosing T_{set} and $time_{set}$ we obtain a trajectory for x_1 and a trajectory for x_2 that are piecewise linear functions. For example if we choose $T_{set} = [k_2, k_3, k_1]$ and $time_{set} = [2000, 500]$ we have the evolutions for x_1 and x_2 as in Figure 4.15. We obtain an admissible trajectory, because $x_2 \leq 7$, with the total duration equal to 10333*. If we choose $T_{set} = [k_4, k_1, k_2]$ and $time_{set} = [2000, 2000]$ we have the evolution for x_1 and x_2 as in Figure 4.16. In this case we obtain a lower total duration equal to* 7000 *but x_2 becomes greater than* 7 *so the trajectory is not respecting the constraint.*

In Figure 4.15 and Figure 4.16 we also plot the real evolution of the thawing process. This linear approximation is trying to respect the time but not the shape of the evolution and it is meant just as a toy example to explain the idea behind the method. Other function, e.g. an exponential function, should be selected to approximate better this phenomenon.

4.2.1 Hybrid Automaton model

A multistage system with n stages is inherently a hybrid model and the natural way to model it is with a hybrid automaton with $n + 1$ locations: n locations for the n stages of the system, plus an extra location for the end condition. Our strategy will consist in increasing (step by step) the number of locations of a hybrid automaton till we reach what we call a *multistage automaton* appropriately modeling our scenario. Based on the notation in Chapter 2 we define a multistage automaton.

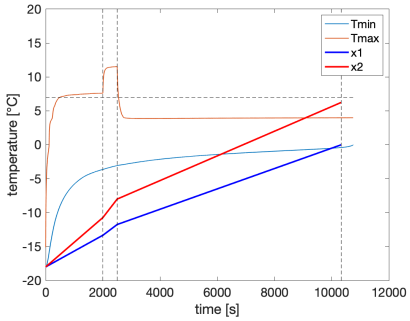


Figure 4.15: Example of evolution with $T_{set} = [k_2, k_3, k_1]$ and $time_{set} = [2000, 500]$. The dashed horizontal line correspond to $7°C$. x_1, x_2 are the linear evolution and T_{min}, T_{max} are the real evolution of the thawing process.

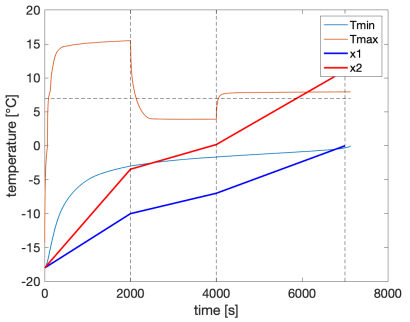


Figure 4.16: Example of evolution with $T_{set} = [k_4, k_1, k_2]$ and $time_{set} = [2000, 2000]$. The dashed horizontal line correspond to $7°C$. x_1, x_2 are the linear evolution and T_{min}, T_{max} are the real evolution of the thawing process.

Definition 4.2.1. *(Multistage automaton)* A multistage automaton $(\mathcal{H}, \mathcal{K}, g)$ *consists of the following components:*

- $\mathcal{K}$ *a finite set of real parameters;*
- $\mathcal{H} = \langle \mathbf{Z}, \mathbf{Z}', \mathcal{V}, \mathcal{E}, Inv, Dyn, Act, Reset \rangle$ *is a hybrid automaton;*
- $\langle \mathcal{V}, \mathcal{E} \rangle$ *is a Direct Acyclic Graph (DAG) and* $\mathcal{V} = S \cup V \cup E$ *a partition of three not empty sets with* S *set of nodes with only outgoing edges,* E *set of nodes with only incoming edges and* V *set of nodes with at least one outgoing edge and at least one incoming edge;*
- $g : V \to \mathcal{K}$ *is a function of labeling;*
- $\mathbf{Z} = (Z_1, \ldots, Z_n) \in \mathbb{R}^n$ *and* $\mathbf{Z}' = (Z'_1, \ldots, Z'_n) \in \mathbb{R}^n$ *are two vectors of variables and there exists a function* $f : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \times \mathcal{K} \to \mathbb{R}^n$ *for each* $v \in V$ *such as*

$$Dyn(v)[\mathbf{Z}, \mathbf{Z}', t] \equiv (\mathbf{Z}' = f(\mathbf{Z}, t, g(v)))$$

We work with the linear multistage system in the Example 4.2.1 to illustrate the associated multistage automaton. Set $\mathcal{K} = \{k_1, k_2, k_3, k_4\} = \{4, 8, 12, 16\}$. If we fix the parameter vector T_{set}, we can define a simple linear hybrid automaton $\mathcal{H}_L = \langle \mathbf{Z}, \mathbf{Z}', \mathcal{V}, \mathcal{E}, Inv, Dyn_f, Act, Reset \rangle$ as in Figure 4.17.

First of all we are going to explicit the role of the variables $T_{set}(i)$. These constants change the evolution of our variables but we have only a finite number of values to take into account, since the dimension of $\mathcal{K}$ is finite.

We split each stage of $\mathcal{H}_L$ in 4 different locations, one for each possible evolution. We add also a dummy starting node to initialize the variables. What we obtain is a

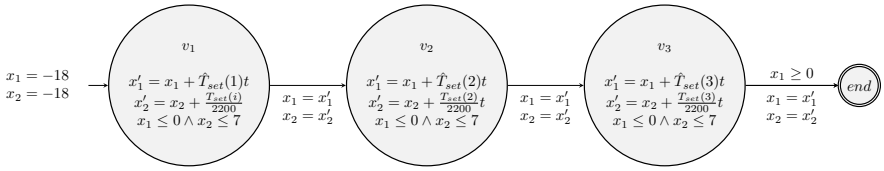


Figure 4.17: Linear Hybrid Automata $\mathcal{H}_L$. We indicate with $\hat{T}_{set}(i)$ the value $\frac{(T_{set}(i)+3.2)}{4800}$. Above each arrow (except the initialization) we indicate the activation condition (if it is no $True$), below we indicate the reset condition.

graph with 3 internal layers as in Figure 4.18. We denote by $v_{i,j}$ a location on layer $i = 1, 2, 3$ with constant $k_j \in \mathcal{K}$. Every location at layer i is connected to all the locations at layer $i + 1$. The layered hybrid automaton $\mathcal{L}$ in Figure 4.18 has:

- $\mathbf{Z} = (x_1, x_2)$, $\mathbf{Z}' = (x_1', x_2')$; $(\mathcal{V}, \mathcal{E})$ as in Figure 4.18;
- $Reset(e)[\mathbf{Z}, \mathbf{Z}'] \equiv (x_1 = x_1' \wedge x_2 = x_2')$ for $e \in \mathcal{E}$;
- $Inv(v_{i,j})[\mathbf{Z}] \equiv (x_1 \leq 0 \wedge x_2 \leq 7)$ for $i = 1, 2, 3$ $j, k = 1, \ldots, 4$; $Inv(v)[\mathbf{Z}] \equiv True$ for $v = s, e$;
- $Dyn(v_{i,j})[\mathbf{Z}, \mathbf{Z}', t] \equiv (x_1' = x_1 + \frac{(k_j+3.2)}{4800}t \wedge x_2' = x_2 + \frac{k_j}{2200}t$ for $i = 1, 2, 3$ $j = 1, \ldots, 4$;
  $Dyn(s)[\mathbf{Z}, \mathbf{Z}', t] \equiv (x_1' = -18 \wedge x_2' = -18)$ and $Dyn(e)[\mathbf{Z}, \mathbf{Z}', t] \equiv (x_1' = x_1 \wedge x_2' = x_2)$;
- $Act((v_{3,j}, e))[\mathbf{Z}'] \equiv (x_1' \geq 0)$ for $j = 1, \ldots, 4$ and the activation is $True$ for all the other edges.

For each internal location $v_{i,j} \in \mathcal{V}$ we can define the function $g(v_{i,j}) = k_j$ with $k_j \in \mathcal{K}$. It is evident by construction that the couple $(\mathcal{L}, \mathcal{K}, g)$ is a multistage automaton.

If we consider the multistage automaton $\mathcal{L}$ in Figure 4.18 and we solve a reachability problem from the location s to the location e we find just a feasible solution for the optimal parameter synthesis but not the optimal one. We need to make a step ahead adding the information of the total duration. In $\mathcal{L}$ we know that the final duration of our system can vary from 1 hour to 4 hours so we can split the location *end* in 4 locations, one per each hour (or more if we want to refine the interval). We obtain the multistage automaton $\bar{\mathcal{L}}$ as in Figure 4.19. We call e_1 the end location associated with $t = 1$ hours and in general e_h for the end location associated with $t = h$ hours.

What we have to modify is the activation constraint adding the time bound:

$$Act((v_{3,j}, e_1))[\mathbf{Z}'] \equiv (x_1' \geq 0 \wedge t \leq 3600);$$

$$Act((v_{3,j}, e_4))[\mathbf{Z}'] \equiv (x_1' \geq 0 \wedge 3 \cdot 3600 < t) \ ;$$

$$Act((v_{3,j}, e_h))[\mathbf{Z}'] \equiv (x_1' \geq 0 \wedge (h-1) \cdot 3600 < t \leq h \cdot 3600)$$

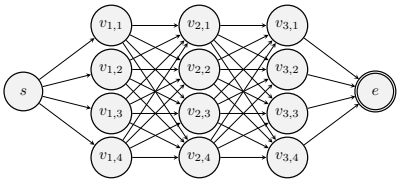


Figure 4.18: Layer Hybrid Automaton $\mathcal{L}$ with 3 layers

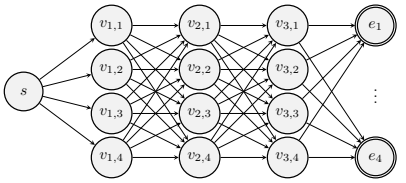


Figure 4.19: Layer Hybrid Automata $\bar{\mathcal{L}}$ with 3 layers

If the reachability problem from the location s with $x_1 = -18°C$ and $x_2 = -18°C$ to the location e_1 have a solution, we have the existence of a feasible solution for the optimal parameter synthesis with a total duration ≤ 1 hour, vice versa if the reachability problem is empty we know that if a solution exists the optimal value is greater than 1 hours.

In general solving the reachability problem on the automaton $\bar{\mathcal{L}}$ give us the existence of a feasible solution for the optimal parameter synthesis problem with a lower/upper bound to the optimal solution.

To conclude we need to be able to solve the reachability problem. In general, if the evolution are linear, we can solve the reachability problem from the set of starting states Σ_S to the set of ending states Σ_E using the backward approach introduced in [6, 5]. The procedure starts with the set $\Sigma_{cur} = \Sigma_E$ and repeatedly adds states from which a state in Σ_{cur} can be reached. The procedure terminates with an affirmative answer if at some stage a state in Σ_S is added, and it terminates with a negative answer if no new states can be added. We know that this procedure ends because we are working with an acyclic graph with trajectories that are limited in time so we have a finite number of steps.

To add states we need to compute two fundamental sets:

- the *precondition* set of a transition $e = (v, v') \in \mathcal{E}$

$$pre_e(\Sigma) = \{\sigma \mid \exists \sigma'(\sigma' \in \Sigma \wedge \sigma \rightarrow_e \sigma')\} \tag{4.10}$$

  with $\sigma = (v, x)$ and $\sigma' = (v', x')$.

- the *backward time closure* set of a location $v \in \mathcal{V}$

$$< \Sigma >_{\swarrow v} = \{\sigma \mid \exists \sigma' \exists t(\sigma' \in \Sigma \wedge t \in \mathbb{R}^{\geq 0} \wedge \sigma \rightarrow_t \sigma')\} \tag{4.11}$$

  with $\sigma = (v, x)$ and $\sigma' = (v, x')$.

Given a formula φ, we write $pre_e(\varphi)$ and $< \varphi >_{\swarrow v}$ to indicate only the set of valuations obtained using the two operators defined above. We can write the recursive Algorithm 1 and if the algorithm terminates without a YES answer we know that the reachability problem has a solution.

Algorithm 4.2 Backtracking Reachability

Input: two locations $v_i \in \Sigma_S$ and $v_f \in \Sigma_E$, a formula $\varphi(x,t)$, a valuation x_i and an acyclic automaton $G = (V, E)$.

function: $backtracking(v_i, v_f, \varphi, x_i, G)$
for $v \in V$ **do**
  if $e = (v, v_f) \in E$ **then**
    if $v == v_i \wedge \varphi(x_i)$ **then**
      Output: YES
    else
      $\varphi_{cur} =< pre_e(\varphi(x,t)) >_{\swarrow v}$
      recursive call $backtracking(v_i, v, \varphi_{cur}, x_i, G)$
    end if
  end if
end for

We can apply the first step of the algorithm to the automaton in Figure 4.19. In this case we have $v_i = s$, $v_f = e_1$, the initial valuation $x_i = (x_1 = -18, x_2 = -18, t = 0)$ and a formula

$$\varphi(x_1, x_2, t) = (x_1 \geq 0 \wedge x_2 \leq 7 \wedge t \leq 3600) \tag{4.12}$$

We have three edges incident in the location $e1$, we can select for example the edge $(v_{3,1}, e1)$ and calculate the formula φ_{cur}.

$$\begin{aligned}
\varphi_{cur} &=< pre[(v_{3,1}, e_1)](\varphi) >_{\swarrow v_{3,1}} \\
&=< x_1 = 0 \wedge x_2 \leq 7 \wedge t \leq 3600 >_{\swarrow v_{3,1}} \\
&= (x_1 + 4 \times 10^{-3} t' = 0 \wedge x_2 + 7.3 \times 10^{-3} t' \leq 7 \wedge t + t' \leq 3600 \wedge x_1 \leq 0 \wedge x_2 \leq 7) \\
&= (-7.3x_1 + 4x_2 \leq 28 \wedge t - 0.25 \times 10^3 x_1 \leq 3600 \wedge x_1 \leq 0 \wedge x_2 \leq 7)
\end{aligned}$$

where t' is a support variable to express the current evolution.

In this case we have linear evolution and we know that both the precondition and the backward time closure set are linear set of valuations. The procedure is effective and we can for example prove that the automaton $\mathcal{L}$ does not have feasible trajectories with a duration less than 2 hours, so the two reachability problems from s to e_1 and e_2 have a negative result.

4.3 Conclusion and future works

New emerging technologies for food service promote the use of advanced analysis methodologies due to a lack of knowledge about the dynamic behavior of the system. Two optimization analyses for a jet impingement multistage thawing process are presented.

The first method is a multi-objective analysis in virtual mode that permits to speed up the development process and permits to make experiments only for the

most promising combinations of parameters, avoiding the time-consuming trial and error process. The implemented methodology is an example of structured approach to optimize a complex parametric process, by combining a validated virtual model with standard optimization techniques. It has also some drawbacks. The price of the reliability and robustness of a GA is paid with the time spent in the tuning process, in particular when quite complex and heavy black box functions are used. Future works are related to the experimental study of some optimal thawing cycles, taken from the obtained Pareto front, by conducting additional food quality analyses.

The second method creates a bridge in this thesis between the optimization techniques and analysis of hybrid systems. Indeed, we showed how we can reformulate the optimal parameter synthesis problem as a reachability problem. With this reformulation we can use the backtrack technique introduced by Alur [6, 5] to prove the existence of a feasible solution, and obtain a lower/upper-bound to an optimal solution. *True* thawing is, in fact, non-linear. The graph construction presented here is applicable to the non-linear case but additional ideas are needed to solve the reachability problem in that case. A further interesting direction consists, therefore, in studying the temperature-evolution in order to approximate its law—to start—with a piece-wise linear function. An option, usually used to find and analytic solution for heating systems without phase change, is the integral method [85]. Additional analysis are need to verify the feasibility of this method for phase changing phenomenon in 3 dimensions.

III

Reachability analysis of hybrid systems

5

Reachable set approximation for robot arms

The paradigm for robot usage has changed in the last few years, from an idea in which robots work with complete autonomy to a scenario where robots cognitively collaborate with human beings. This brings together the best of each partner, robot and human, by combining coordination, dexterity and cognitive capabilities of humans with the robots' accuracy, agility and ability to sustain repetitive work.

The most crucial problem that must be solved in order to enable this paradigm is to ensure a safe collaboration, i.e. obstacle avoidance. The obstacle avoidance is literally a reachability analysis. We want to physically reach an object with the robot arm, while avoiding some obstacles.

In the pioneering work of Khatib [65], a real-time obstacle avoidance approach based on the classical artificial potential field concept is introduced. After this work a wide variety of approaches has been proposed to accomplish the obstacle avoidance challenge [96, 95, 42, 89].

We can distinguish between two types of obstacles: fixed obstacles and movable obstacles. To solve the complete obstacle avoidance problem, both have to be accounted for. While with moving obstacles we are forced to use real-time algorithms, a key point with fixed obstacles is the definition of the reachable space of robots. If we are able to reflect the presence of the fixed obstacle in the definition of the workspace we can light the real-time work and support the decision process.

What we propose is to formally describe the obstacle avoidance as a reachability problem and use a technique from the verification field to estimate the reachability space. Indeed a robot is, in a natural way, a trigonometric dynamical system, we will see in the following the precise expression of the evolution law. The field of dynamical systems has provided us various tools to design the reachable region for nonlinear dynamical systems. Reachable set definition through Bernstein coefficients is one of these techniques [33]. This approach allows not only to describe the reachable space but, in principle, to certificate that the robot will not reach a given region.

The definition of reachable space in robotic field is a mature topic that has received much attention during the last decades The majority of the methods used to define this region can be classified into three groups: geometrical methods, dis-

cretization/sampling methods, and singularity-based methods. Geometrical methods can handle kinematic constraints such as joint limits and even self-collisions, but only in relatively simple cases [29]. Sampling methods generate many configurations of the robot, and check if each configuration belongs to the workspace satisfying all kinematic constraints [10, 113]. Singularity-based methods works with kinematic constraints written as equalities and, in general, they solve the problem numerically [50, 19].

Among these methods, only few describe a direct relationship between the workspace and the joint space, where we formulate the trajectory. For example in [100] a task constraints method obtained by sampling the positions in the joint space is described. In [109] instead, using the traditional ellipsoids kinematic analysis, they extend the reachable space with a quality index that penalizes the behaviors near the joint limits and near a possible obstacle. Unfortunately in [43] they have already shown that in the general case the polytopes approximation provides more accurate estimations with respect to the ellipsoids method.

In this work we want to use the Bernstein theory to approximate the reachable space with polytopes given a limitation on the joint space. Moreover we want to use this approximation to define joint constraints when fixed obstacle are present.

5.1 Forward kinematics of a robot arm

A robot arm is an assembly of rigid parts (*links*) and moving parts (*joints*) that connect two different links. A joint is a moving element with constraints on the directions. There are two main types of joints:

1. prismatic joint: translation movement along one axis;
2. rotational joint (or revolute): rotation movement around one axis.

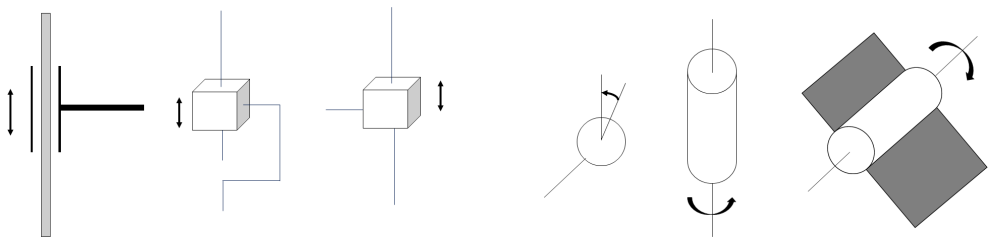


Figure 5.1: Joint types: 3 prismatic joints (left) and 3 revolute joints (right)

A *kinematic chain* robot is a robot defined with n joints and $n+1$ links and the joints and links are alternated. Each link is connected with only other two rigid bodies, except for the first and the last links that are connected only with one rigid body.

Given a kinematic chain robot with n joints and given the joint variable $\theta = (\theta_1, \ldots, \theta_n)$ we define the *joint space* as the set of joint vectors $\Theta \subset \mathbb{R}^n$. We define also the *work space* as the subset $\mathbb{P} \in \mathbb{R}^3$ of reachable space positions.

The *forward kinematics* is the problem of moving from the joint space Θ to the working space $\mathbb{P}$. A standard way to solve the forward kinematics problem is to define the transformation T.

$$T : \Theta \to \mathbb{P} \times \{1\}$$

such as $\theta \in \Theta$ and $T(\theta) = P \in \mathbb{P} \times \{1\}$ with $P = (x, y, z, 1)$.

The *inverse kinematics* instead is the problem to find one of the possible inverse transformation to map a point in the working space to a configuration of the robot in the joint space. This reverse operation is more challenging and generally a robot arm is built to have multiple configurations in the internal part of the working space.

Lets focus on the forward kinematics and we want to define the transformation T. By working in the space $\mathbb{P} \times \{1\}$, we are able to use a 4×4 matrix for describing rotation and translation movements. In this space the composition is just a matrix product.

We will describe T for a specific robot but the construction is general for each kinematic chain robot. We are working with the Comau robot Racer5-0.63. This robot has 6 revolute joints, $\theta_1, \ldots, \theta_6$, and 7 links (Figure 5.2). The joint space is a

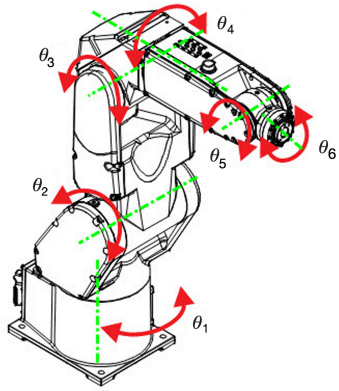


Figure 5.2: Comau robot Racer5-0.63

Table 5.1: Joint limitations for Comau robot Racer5-0.63

	min [°]	max [°]
θ_1	-170	170
θ_2	-95	135
θ_3	-90	155
θ_4	-200	200
θ_5	-125	125
θ_6	-2700	2700

subset of $\mathbb{R}^6$ defined with bounds in Table 5.1. We define with the positive direction of θ_2 e θ_3 the frontal direction of the robot. If we move 90° the joint θ_2 the robot is 90° bending forward, if we move $-90°$ the robot is 90° bending backward.

In order to define the transformation T we need to define the coordinate systems as in Figure 5.3. The main coordinate systems are the followings:

- World frame: coordinate system of the environment in which the robot is installed.
- Base frame: coordinate system of the robot base.
- Uframe: coordinate system of the working surface (e.g. a table or a desk).
- Tool frame: coordinate system of the central point in which we can attach a tool.

In the following we assume that the word frame is the same as the base frame.

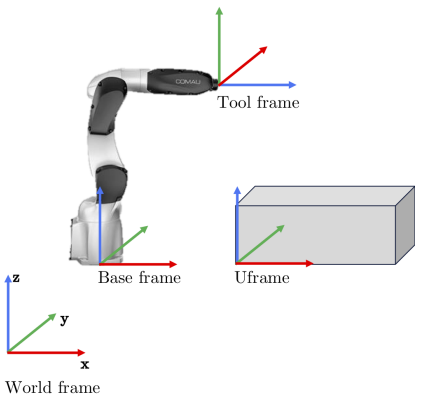


Figure 5.3: Coordinate systems for Comau robot Racer5-0.63

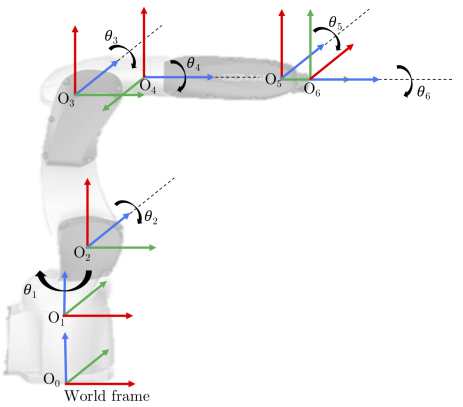


Figure 5.4: Transformation from tool frame to world frame

The transformation T can be created defining a transformation chain that translates the coordinates of a point from the tool frame to the world frame (Figure 5.4). In particular we want to express the position of the origin of the tool frame with respect to the world frame. In the figure we can see that for moving from the origin O_1 to the origin O_0 we need to use only a translation along z axis. Instead from origin O_2 to origin O_1 we need a translation and a rotation. The transformation T is a function of the joint variables $\theta_1, \ldots, \theta_6$. T allows us to express the point O_6 with respect to the world frame.

The transition from the coordinate system of the rigid body i to the coordinate system of the rigid body j can be described using the homogeneous transformation T_i^j

$$T_i^j = \begin{bmatrix} M_i^j & v_i^j \\ 0\,0\,0 & 1 \end{bmatrix} \tag{5.1}$$

with $M \in \mathbb{R}^{3\times3}$ a rotation matrix and $v \in \mathbb{R}^{3\times1}$ a translation vector.

T is the matrix product of n homogeneous transformations, from one link to the next.

$$T = \prod_{i=1}^{n} T_i^{i+1} \tag{5.2}$$

Every homogeneous transformation T_i^j is the product of two different matrices: a constant homogeneous transformation A_i^j, that describes the geometric property of the link, and a rotation matrix $R_i^j(\theta)$ around z axis that is the movement of the revolute joint θ_i. We assume that the rotation axis (and translation axis) is always the z axis of the reference frame of the next link. For example θ_1 is a joint between link 1 and link 2 and it describes a rotation of the z axis of the coordinate system of

the link 2. The matrix $R_i^j(\theta)$ is:

$$R_i^j(\theta_i) = \begin{bmatrix} \cos(\theta_i) & \sin(\theta_i) & 0 & 0 \\ -\sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

In general $T_i^j = R_i^j(\theta_i) \cdot A_i^j$, then $T(\theta) = \prod_{i=1}^n R_i^{i+1}(\theta_i) \cdot A_i^{i+1}$. We need also to add another matrix A_0^1 to transform the base frame (reference frame of the first link) to the coordinate system of the second link. The final expression for T is

$$T(\theta) = A_0^1 \cdot \prod_{i=1}^{n} R_i^{i+1}(\theta_i) \cdot A_i^{i+1} \tag{5.4}$$

Example 5.1.1. *In the case of the Racer*5-0.63*, if we want to move from O_0 to O_1 we need to translate along z of* 24.2*cm by defining the matrix A_0^1*

$$A_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 24.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then for moving from the first link to the second we have

$$T_1^2(\theta_1) = \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & 0 & 0 \\ -\sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 12.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation $T(\theta)$ returns the vector $(x, y, z, 1)$. We define with $t_x(\theta)$, $t_y(\theta)$ and $t_z(\theta)$ the three functions associated to $T(\theta)$ that return the three coordinates.

$$\begin{aligned} x &= T(\theta)(1) =: t_x(\theta) \\ y &= T(\theta)(2) =: t_y(\theta) \\ z &= T(\theta)(3) =: t_z(\theta) \\ 1 &= T(\theta)(4) \end{aligned} \tag{5.5}$$

5.1.1 Adding a tool

Almost all the robots allow the installation of a tool connected to the last link. On the tool we can define the *Tool Central Point* (TCP), a characteristic point that represents the center of the utensil. We can define a new coordinate system with the origin on the TCP and the orientation that agrees with the tool frame. Given the translation vector d_{TCP} from the origin of the tool frame to the TCP, we want to perform the forward kinematic of the TCP. Given the vector θ, we can follow two ways that are equivalent.

- We apply the transformation T defined in Eq. 5.4 to the point (d_{TCP},1), that is the TCP point with respect to the tool frame.

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = T(\theta) \cdot \begin{pmatrix} d_{TCP}(1) \\ d_{TCP}(2) \\ d_{TCP}(3) \\ 1 \end{pmatrix} \tag{5.6}$$

- We define a new coordinate system on the TCP point and we treat the tool as a extra link connected with a fixed joint.

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = T(\theta) \cdot \begin{bmatrix} 1 & 0 & 0 & d_{TCP}(1) \\ 0 & 1 & 0 & d_{TCP}(2) \\ 0 & 0 & 1 & d_{TCP}(3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.7}$$

In the following we will use the approach of Equation 5.7 and we will define the constant matrix A_{n+1}^{n+2} that is the translation matrix from the origin of the tool frame to the TCP point. We are considering the tool as the $n+2$ link with a fixed joint then the matrix R_{n+1}^{n+2} is the identity.

Our robot Racer5-0.63 has 6 joints, 7 links and a shovel-shaped cooking tool (Figure 5.5) connected to the end of the link 7. The TCP point is on the center of the external edge on the flat part of the tool. In the following we use $T(\theta)$ to indicate the transformation matrix of a point from the coordinate system of the TCP to the world frame.

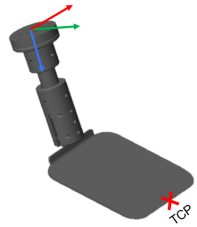


Figure 5.5: shovel-shaped cooking tool connected to the Racer5-0.63

5.2 Reachability region

Given a joint combination $\theta \in \Theta$ we obtain the point $T(\theta)$ in the work space $\mathbb{P} \times \{1\}$. All the points in the work space are reachable points. Starting from a position $\theta = (\theta_1, \ldots, \theta_n)$ and adding a limitation $\underline{\epsilon} = (\underline{\epsilon_1}, \ldots, \underline{\epsilon_n})$ and $\overline{\epsilon} = (\overline{\epsilon_1}, \ldots, \overline{\epsilon_n})$ to each joint we want to describe the reachability region as

$$Reach(\theta, \underline{\epsilon}, \overline{\epsilon}) = \{p \in \mathbb{P} | \forall i \in \{1, \ldots, n\} \exists \delta_i \in [-\underline{\epsilon_i}, \overline{\epsilon_i}] : T(\theta + \delta) = (p, 1)\} \tag{5.8}$$

Lets assume, in first approximation, to work with a symmetric interval with $\epsilon = \underline{\epsilon} = \overline{\epsilon}$. The problem of computing reachability region from an initial condition is call *reachability problem*.

If T is the forward kinematic function for the TCP of the robot, the starting position $P \in \mathbb{P} \times \{1\}$ is the point

$$P = T(\theta) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = A_0^1 \cdot \left(\prod_{i=1}^{n} R_i^{i+1}(\theta_i) \cdot A_i^{i+1} \right) A_{n+1}^{n+2} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

while a generic point in the reachability region is described by

$$\begin{aligned} P' =& T(\theta + \delta) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = A_0^1 \cdot \left(\prod_{i=1}^{n} R_i^{i+1}(\theta_i + \delta_i) \cdot A_i^{i+1} \right) A_{n+1}^{n+2} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\ =& A_0^1 \cdot \begin{bmatrix} \cos(\theta_1 + \delta_1) & \sin(\theta_1 + \delta_1) & 0 & 0 \\ -\sin(\theta_1 + \delta_1) & \cos(\theta_1 + \delta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot A_1^2 \cdot \dots \\ & \cdot \begin{bmatrix} \cos(\theta_n + \delta_n) & \sin(\theta_n + \delta_n) & 0 & 0 \\ -\sin(\theta_n + \delta_n) & \cos(\theta_n + \delta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot A_n^{n+1} \cdot A_{n+1}^{n+2} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

Since the rotation matrix of an angle $\theta + \delta$ is the product of the rotation matrix of the angle θ and the rotation matrix of the angle δ, we can rewrite the point P' as

$$\begin{aligned} P' =& T(\theta + \delta) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \\ =& A_0^1 \cdot \begin{bmatrix} \cos(\theta_1) & \sin(\theta_1) & 0 & 0 \\ -\sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\delta_1) & \sin(\delta_1) & 0 & 0 \\ -\sin(\delta_1) & \cos(\delta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \\ & A_1^2 \cdot \ \dots \ \cdot \begin{bmatrix} \cos(\theta_6) & \sin(\theta_6) & 0 & 0 \\ -\sin(\theta_6) & \cos(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\delta_n) & \sin(\delta_n) & 0 & 0 \\ -\sin(\delta_n) & \cos(\delta_n) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \\ & A_n^{n+1} \cdot A_{n+1}^{n+2} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = A_0^1 \cdot \left(\prod_{i=1}^{n} R_i^{i+1}(\theta_i) \cdot R_i^{i+1}(\delta_i) \cdot A_i^{i+1} \right) \cdot A_{n+1}^{n+2} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

By changing δ inside the interval $[-\epsilon_i, \epsilon_i]$ we obtain the reachability region. This region has 2^n vertices. We can calculate each vertex setting $\delta_i = \epsilon_i$ or $\delta_i = -\epsilon_i$.

Example 5.2.1. *In Figure 5.6 we can see an example of reachability region for the robot Racer*5-0.63*. The red dot is the starting position that correspond to* $\theta = (0°, -90°, 0°, 0°, 0°, 0°)$*. The black dots are all the vertices of the region. The lines are obtained by selecting* $j \in \{1, \dots, 6\}$*, fixing* $\delta_i = \pm\epsilon_i$ *for all* $i \neq j$ *and varying just* $\delta_j \in [-\epsilon_j, \epsilon_j]$*. We have set* $\epsilon_i = 5°$ *for* $i \in \{1, \dots, 6\}$*.*

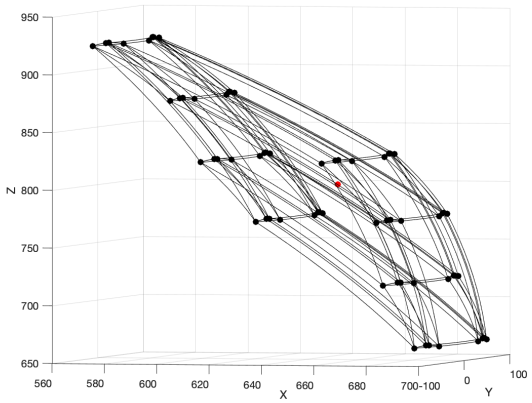


Figure 5.6: Example of a reachability region for the robot Racer5-0.63

We can also consider the forward kinematic function T for other critical points like the wrist or the elbow of the robot. Being able to solve this problem for different points is fundamental to analyze the behavior of the robot and to address safety-critical scenarios.

Here we want to address the problem to avoid a fixed obstacle. We consider a *red zone* $\mathcal{R}$ that we want to avoid. This zone is a polytope defined with a set of linear inequalities.

$$\mathcal{R} := \{x \in \mathbb{R}^3 | A_i x \leq b_i \quad \forall i = 1, \dots, m\} \tag{5.9}$$

with $A_i \in \mathbb{R}^3$ and $b_i \in \mathbb{R}$.

We want to solve these two problems:

1. *Safety problem*: decide if the reachability region crosses the red zone;

2. *Max-safety problem*: define the maximum intervals $[-\underline{\epsilon_i}, +\overline{\epsilon_i}]$ such as the reachability region does not intersect the red zone.

In the following we will address these problems with respect to the TCP, but we can make the same analysis for any critical point, by changing the forward kinematic function T.

5.3 Safety problem

In the safety problem the variables θ and ϵ are fixed, so the reachability region is fixed too. The coordinates of an arbitrary point in the reachability region, with respect to the world frame, are defined by

$$\begin{aligned} x &= T(\theta+\delta)(1) =: t_x(\delta) \\ y &= T(\theta+\delta)(2) =: t_y(\delta) \\ z &= T(\theta+\delta)(3) =: t_z(\delta) \end{aligned}$$

where only δ is a variable.

The safety problem can be solved with the following feasibility problem.

$$\begin{aligned} A_{i,1}t_x(\theta+\delta) + A_{i,2}t_y(\theta+\delta) + A_{i,3}t_z(\theta+\delta) \leq b_i \qquad & \forall j = 1,\dots,m \\ -\epsilon_i \leq \delta_i \leq \epsilon_i \qquad & \forall i = 1,\dots,n \end{aligned} \tag{5.10}$$

If the feasibility problem has a solution δ^* then the reachability region crosses the red region $\mathcal{R}$ and the solution $\theta + \delta^*$ is a point in the intersection. If the feasibility problem is unfeasible, the two regions have an empty intersection.

The three functions t_x, t_y and t_z are trigonometric functions. We can decide to solve directly the problem by calling a non-linear optimization solver, or we can work on the expression of the reachability region.

Inspired by the work of Dreossi [34] we follow this second strategy and we overapproximate the reachability region using the Bernstein theory for polynomial function (see Chapter 2.2). First of all we need to transform the three functions in a polynomial. We define two vectors, X and Y, of n variables, such as $Xi := \cos(\delta_i)$ and $Yi := \sin(\delta_i)$. We denote with p_x, p_y and p_z the three functions obtained replacing each occurrence of $\cos(\delta_i)$ and $\sin(\delta_i)$ with Xi and Yi.

$$\begin{aligned} x &= T(\theta+\delta)(1) = t_x(\delta) =: p_x(X,Y) \\ y &= T(\theta+\delta)(2) = t_y(\delta) =: p_x(X,Y) \\ z &= T(\theta+\delta)(3) = t_z(\delta) =: p_x(X,Y) \end{aligned}$$

Now p_x, p_y and p_z are polynomial functions in $2n$ variables. We need also to add n trigonometric constraints, so the new feasibility problem is

$$\begin{aligned} A_{i,1}p_x(X,Y) + A_{i,2}p_y(X,Y) + A_{i,3}p_z(X,Y) \leq b_i \qquad & \forall j = 1,\dots,m \\ Xi^2 + Yi^2 = 1 \qquad & \forall i = 1,\dots,n \\ \cos(\epsilon_i) \leq Xi \leq 1 \qquad & \forall i = 1,\dots,n \\ -\sin(\epsilon_i) \leq Yi \leq \sin(\epsilon_i) \qquad & \forall i = 1,\dots,n \end{aligned} \tag{5.11}$$

We can observe that the three functions are polynomial functions of degree n in $2n$ variables. This is true independently of the value of θ because it is a direct consequence of the way in which we have built the transformation T and depends only on the fact that we are working with a kinematic chain robot.

5.3.1 Overapproximation of the reachability region

The reformulation of the functions t_x, t_y and t_z in polynomial form allowed us to use the overapproximation using the Bernstein polynomial. We follow the steps of [34] in order to create a parallelotope that encloses the reachability region.

The three functions p_x, p_y and p_z are polynomials $p(X,Y) = p(x) \in \mathbb{R}^{2n} \to \mathbb{R}$ with the first n variables that represents $\cos(\delta)$ and the last n variables that represents $\sin(\delta)$. Every polynomial can be represented using the power basis as follows:

$$p(x) = \sum_{i \in I^n} a_i x^i \tag{5.12}$$

where the index $i = (i_1, \ldots, i_{2n})$ is a multi-index of size $2n \in \mathbb{N}$ and x^i denotes the monomial $x_1^{i_1} x_2^{i_2} \ldots x_{2n}^{i_{2n}}$.

As a direct consequence of creating T as a matrix product of rotations, we can observe that the multi-index i of our polynomial $p(x)$ satisfy the following properties.

Property 5.1.

1. $\sum_{j=1,\ldots,2n} i_j \leq n$
2. $i \in \{0,1\}^{2n}$
3. $i_j + i_{n+j} = 1$

We want to represent $p(x)$ using Bernstein basis. Following the definitions in Chapter 2.2 we define the *Bernstein coefficient* for a polynomial $p(x)$ as

$$b_i(p) = \sum_{j \leq i} \frac{\binom{i}{j}}{\binom{deg}{j}} a_j \tag{5.13}$$

Thanks to the second point in Property 5.1 we have $j \in \{0,1\}^{2n}$ and $deg = (1, \ldots, 1)$. The binomial coefficients are all equals to 1 and also the fraction is 1. In our case we obtain:

$$b_i(p) = \sum_{j \leq i} a_j \tag{5.14}$$

We can use the Sharpness Property 2.1 to compute all of them, because we have $\mathcal{V}_{deg} = [0,1]^{2n}$. The vertices are all the possible 2^{2n} multi-index $i \in \{0,1\}^{2n}$.

To overapproximate the reachability region we try the usage of both the range enclosing Property 2.2 and the convex hull property 2.3. These properties are true on the unitary box, but can be generalized over an arbitrary box.

Our domain is defined by sine and cosine of ϵ_i. Given $\epsilon_i \in [0, 90°]$ one possible domain for the variables X, Y is $D1 \times D2$ with

$$\begin{aligned} D1 :=& [\cos(\epsilon_1), 1] \times \ldots [\cos(\epsilon_n), 1] \\ D2 :=& [-\sin(\epsilon_1), \sin(\epsilon_1)] \times \ldots [-\sin(\epsilon_n), \sin(\epsilon_n)] \end{aligned} \tag{5.15}$$

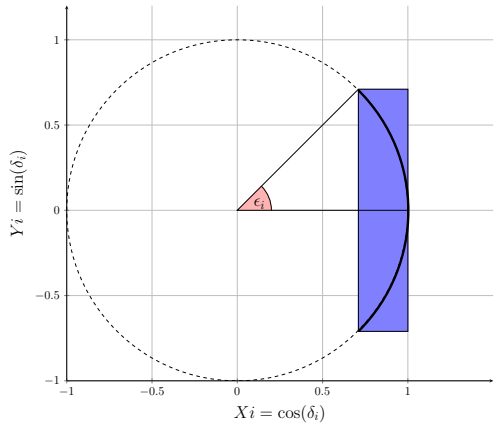


Figure 5.7: Rectangular domain for two couple of variables (Xi, Yi)

In Figure 5.7 we have a graphical representation of the domain $[\cos(\epsilon_i), 1] \times [-\sin(\epsilon_i), \sin(\epsilon_i)]$ for a couple of variable (Xi, Yi) associated to the angle variable $\delta_i \in [-\epsilon_i, \epsilon_i]$

We define the transformation $v(x)$ that maps the unitary box $[0, 1]^{2n}$ in the domain $D1 \times D2$:

$$v(x) = \begin{bmatrix} 1-\cos(\epsilon_1) & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 1-\cos(\epsilon_n) & 0 & \dots & 0 \\ 0 & \dots & 0 & 2\sin(\epsilon_1) & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 2\sin(\epsilon_n) \end{bmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{2n} \end{pmatrix} + \begin{pmatrix} \cos(\epsilon_1) \\ \vdots \\ \cos(\epsilon_n) \\ -\sin(\epsilon_1) \\ \vdots \\ -\sin(\epsilon_n) \end{pmatrix}$$

We can observe that $p(v(x))$ is still a polynomial, so we can calculate all the Bernstein coefficients $b_i^v(p) = p(v(i/deg))$. In the end we are computing all the Bernstein coefficients kust applying the polynomial function on the vertices of the box domain $D1 \times D2$.

Range enclosing property

Lets work directly with an example. For the Racer5-0.63 we have 2^{12} vertices in $\mathcal{V}_{deg}$ so we have 2^{12} Bernstein coefficients for each function p_x, p_y and p_z. For each function we can find the minimum and the maximum coefficient. For example if we consider p_x we can calculate the bounds:

$$X_{\min} = \min_{i \in I^{2n}} p_x(v(i/deg)) \leq p_x(v(x)) \leq \max_{i \in I^{2n}} p_x(v(i/deg)) = X_{\max}$$

The reachability region is inside the intersection of the two half-spaces $X \geq X_{\min}$ and $X \leq X_{\max}$.

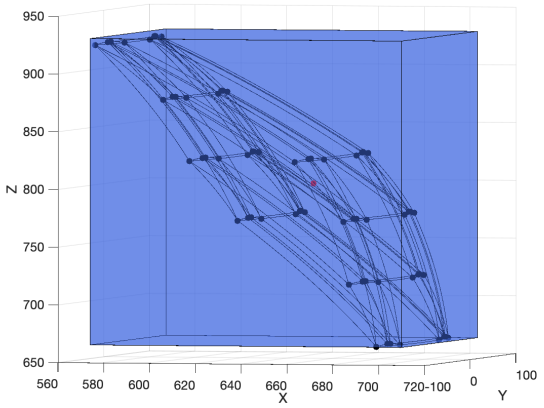


Figure 5.8: Overapproximation of a reachability region with an hyperrectangle.

We have to repeat the same operation for the other two functions and we obtain a parallelepiped that contains the reachability region (Figure 5.8).

The obtained overapproximation can be refined with two different approaches:

1. by modifying the hyperrectangle that defines the domain (e.g. using an hyperparallelogram);
2. by modifying the final directions, that is, instead of overapproximate the direction x, y and z we overapproximate a linear combination of that directions $ax + by + cz$.

We describe these two approaches.

1. Different overapproximation by changing the domain. The domain $D1 \times D2$ is obtained using an upperbound and a lowerbound to Xi and Yi. In Figure 5.7 we can see the arc that defines the relationship between these two variables, that is $Xi = \cos(\delta_i)$ and $Yi = \sin(\delta_i)$ with $\delta_i \in [-\epsilon_i, \epsilon_i]$. If we consider only a couple of variables (Xi, Yi), the domain is the blue rectangle in Figure 5.7.

Now we want to substitute the rectangle with a parallelogram, as the green parallelogram in Figure 5.9 obtained using two tangent lines.

A parallelogram in $2n$ dimensions can be described using the generator representation. In this representation we have to specify a vector $q \in \mathbb{R}^{2n}$ called *vertex* and $2n$ vectors g^i called *generators*. Given q, g^1 and g^2 as in Figure 5.10 we can describe the parallelogram B as follows:

$$B = \{q + \alpha_1 g^1 + \alpha_2 g^2 | (\alpha_1, \alpha_2) \in [0, 1]^2\} \tag{5.16}$$

Then we define the function $\bar{v}(x)$ that maps the unitary square into the parallelogram B:

$$\bar{v}(x) = q + \sum_{i=1}^{2n} g^i x(i) \tag{5.17}$$

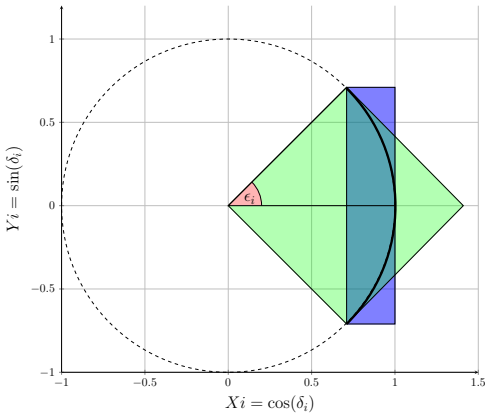


Figure 5.9: Rectangular domains (blue) and parallelogram domain (green) for the couple (Xi, Yi)

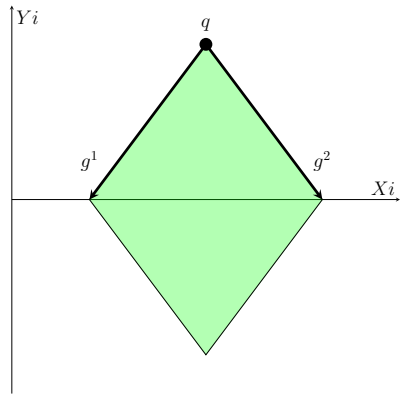


Figure 5.10: Parallelogram with vertex and generators.

Finally we calculate the Bernstein coefficients and the relative bounds of the new polynomial $p(\bar{v}(x))$.

It's possible to use the same representation also for describing the domain $D1 \times D2$. For this domain we have, for each couple of variables (Xi, Yi), $q = (\cos(\epsilon_i), \sin(\epsilon_i))$, $g^1 = (1 - \cos(\epsilon_i), 0)$ and $g^2 = (0, -2\sin(\epsilon_i)$.

$$v(x) = \begin{pmatrix} \cos(\epsilon_1) \\ \vdots \\ \cos(\epsilon_n) \\ \sin(\epsilon_1) \\ \vdots \\ \sin(\epsilon_n) \end{pmatrix} + \sum_{i=1}^{n} \begin{pmatrix} 1 - \cos(\epsilon_1) \\ \vdots \\ 1 - \cos(\epsilon_n) \\ 0 \\ \vdots \\ 0 \end{pmatrix} x(i) + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -\sin(\epsilon_1)) \\ \vdots \\ -\sin(\epsilon_n) \end{pmatrix} x(i+n) \tag{5.18}$$

Example 5.3.1. *We are still working on the same example, with* $\theta = (0°, -90°, 0°, 0°, 0°, 0°)$ *and* $\epsilon_i = 5°$ *for each* $i \in \{1, \ldots, n\}$. *We create a parallelogram using the tangent lines to the point that correspond to the angle at* $5°$ *and to the angle at* $-5°$. *For each couple* (Xi, Yi) *we create a parallelogram like the parallelogram* B *in Figure 5.10 and we define it following the Equation 5.16. In this case we have* $q = (\cos(5°), \sin(5°))$, $g^1 = (\tan(5°)\sin(5°), -\sin(5°))$ *and* $g^2 = (-\tan(5°)\sin(5°), -\sin(5°))$. *Then we build the hyperparallelogram* $b^{\bar{v}} \in \mathbb{R}^{2n}$ *and we create the function* $\bar{v} : [0,1]^{2n} \to B$ *as follows:*

$$\bar{v}(x) = \begin{pmatrix} \cos(5°) \\ \vdots \\ \cos(5°) \\ \sin(5°) \\ \vdots \\ \sin(5°) \end{pmatrix} + \sum_{i=1}^{n} \begin{pmatrix} \tan(5°)\sin(5°) \\ \vdots \\ \tan(5°)\sin(5°) \\ -\sin(5°) \\ \vdots \\ -\sin(5°) \end{pmatrix} x(i) + \begin{pmatrix} -\tan(5°)\sin(5°) \\ \vdots \\ -\tan(5°)\sin(5°) \\ -\sin(5°) \\ \vdots \\ -\sin(5°) \end{pmatrix} x(i+n)$$

We are now ready to calculate the Bernstein coefficients of $p_x(\bar{v}(x))$, $p_y(\bar{v}(x))$ *and* $p_z(\bar{v}(x))$. *We have new upperbounds and lowerbounds that create the green parallelepiped in Figure 5.11. The blue parallelepiped is the one created starting from the hyperrectangle* $D1 \times D2$ *using the function* $v(x)$.

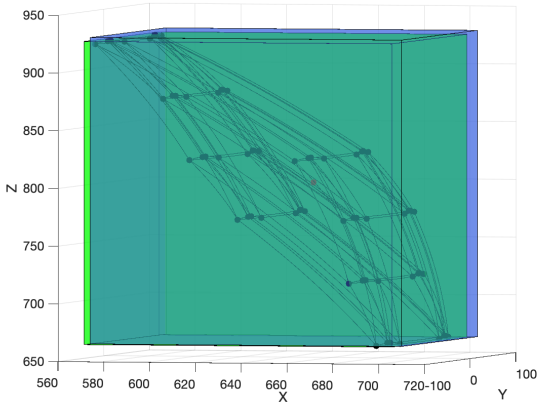


Figure 5.11: Two overapproximation with parallelepiped (green and blue) changing the domain.

2. Different overapproximation by changing directions. Given a domain, we can use the hyperrectangle $D1 \times D2$, we want to work on the shape of the overapproximation region, so on the parallelepiped. In particular we want to change the directions of the planes that we are using.

Given the polynomial $p_x(v(x))$ we calculate the maximum Bernstein coefficient $b^v_{\max}(p_x)$ and the minimum Bernstein coefficient $b^v_{\min}(p_x)$. With these two values we can define two planes $x = b^v_{\max}(p_x)$ and $x = b^v_{\min(p_x)}$ and we know that the reachability region is inside these two planes. We can consider the planes $ax + by + cz = k_{\min}$ and $ax + by + cz = k_{\max}$. Again we have that $ap_x(v(x)) + bp_y(v(x)) + cp_z(v(x))$ is a polynomial and we can overapproximate it using Bernstein coefficients.

Example 5.3.2. *Given* $\theta = (0°, -90°, 0°, 0°, 0°, 0°)$ *and* $\epsilon_i = 5°$ *with* $i \in \{1, \dots, n\}$ *we consider the hyperrectangle* $D1 \times D2 = [\cos(5°), 1]^n \times [-\sin(5°), \sin(5°)]^n$. *We want to overapproximate the reachability region using the plane* $2x + z = k$. *We have to calculate the maximum Bernstein coefficient and the minimum Bernstein coefficient of the polynomial* $2p_x(v(x)) + p_z(v(x))$. *We obtain two planes that are the green planes in Figure 5.12.*

These approaches can be merged to improve the final approximation. We can collect the Bernstein coefficients over different input domain and compute the bounds for a set of possible output directions. We can define a set k of directions $L \in \mathbb{R}^{k \times 3}$ such as $L_i x = k$ define a plane. For example we can define all the directions in Figure

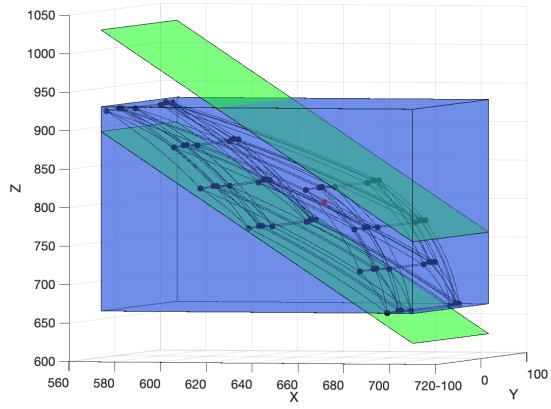


Figure 5.12: Overapproximation of the reachability region using the plane $2x + z = k$.

5.12.

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 0 & 1 \end{pmatrix}$$

In Figure 5.13 we plot the approximation obtained considering the directions L and the two different domains in Figure 5.9.

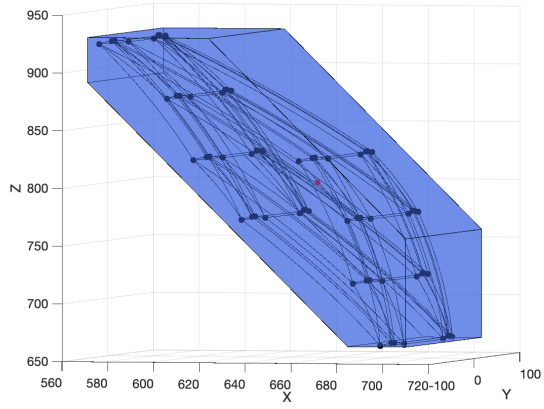


Figure 5.13: Overapproximation of the rechability region obtained by taking the intersection of different input domains and different output directions.

Convex hull property

The range enclosing property create a less precise domain but allows to control the shape of the final overapproximation. The described techniques is useful in general when we need to create a pipe flow of reachable regions. If we are interested, as in this case, just on the single precise overapproximation then we can use Property 2.3 considering the convex hull of all Bernstein coefficients. Thanks to [15] we have that this approximation is exactly the image of the box domain. In Figure 5.14 we plot the obtained output. We can refine it by changing domain as we describe above. Considering the two domains in used for Example 5.3.1 we obtain the two convex hull in Figure 5.15.

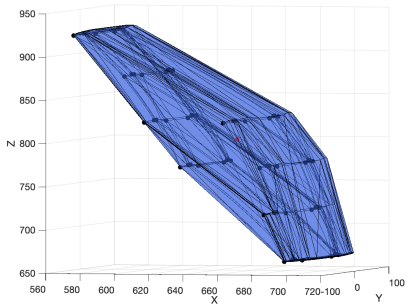


Figure 5.14: Overapproximation of a reachability region with convex hull using domain $D1 \times D2$.

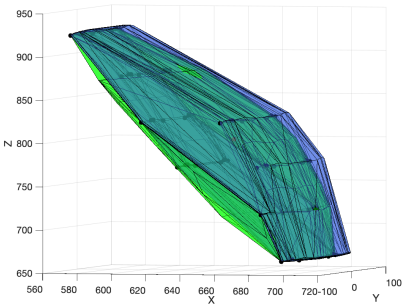


Figure 5.15: Two overapproximation of a reachability region with convex hull using two different domains (green and blue).

As a final consideration we also observe that we have worked with a symmetrical interval $[-\epsilon_i, \epsilon_i]$ but all the described procedures can easily be extended on a general interval, by changing the definition of the input domain.

5.4 Max-safety problem

In the max-safety problem we want to maximize the intervals $[-\underline{\epsilon_i}, +\overline{\epsilon_i}]$ ensuring that the reachability zone does not intersect the red zone. We consider again a symmetric interval with $\epsilon_i = \underline{\epsilon_i} = \overline{\epsilon_i}$. We also assume that the starting position of the robot is outside the red zone, so the problem is not trivial.

We remember that the polynomial $p(x)$ is a polynomial of degree n in $2n$ variables such as the first half of variables $x(1), \ldots, x(n)$ are the variables $Xi = \cos(\delta_i)$ and the second half of variables $x(n+1), \ldots, x(2n)$ are the variables $Yi = \sin(\delta_i)$ with $\delta_i \in [-\epsilon_i, \epsilon_i]$. In the following we will consider $\epsilon_i \in [0, 90°]$ and we choose for our variables X, Y the domain $D1 \times D2$ that we have already defined in Equation 5.15.

We can take again the function $v(x)$ to move from the unitary box to the current domain, choosing the expression of Equation 5.18. In this problem ϵ_i is a variable. We define two new families of variables: $\tilde{X}i = \cos(\epsilon_i)$ and $\tilde{Y}i = \sin(\epsilon_i)$. We define the

vector $\tilde{x}$ such as the first half of variables $\tilde{x}(1), \ldots, \tilde{x}(n)$ are the variables $\tilde{X}i = \cos(\epsilon_i)$ and the second half of variables $\tilde{x}(n+1), \ldots, \tilde{x}(2n)$ are the variables $\tilde{Y}i = \sin(\epsilon_i)$. We rewrite the Equation 5.18 as follows

$$\begin{aligned} v(x,\tilde{x}) = & \begin{pmatrix} \tilde{X}1 \\ \vdots \\ \tilde{X}n \\ \tilde{Y}1 \\ \vdots \\ \tilde{Y}n \end{pmatrix} + \sum_{i=1}^{n} \begin{pmatrix} 1-\tilde{X}1 \\ \vdots \\ 1-\tilde{X}n \\ 0 \\ \vdots \\ 0 \end{pmatrix} x(i) + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -\tilde{Y}1 \\ \vdots \\ -\tilde{Y}n \end{pmatrix} x(i+n) \\ = & \tilde{x} + \sum_{i=1}^{n} \begin{pmatrix} 1-\tilde{x}(1) \\ \vdots \\ 1-\tilde{x}(n) \\ 0 \\ \vdots \\ 0 \end{pmatrix} x(i) + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -\tilde{x}(n+1) \\ \vdots \\ -\tilde{x}(2n) \end{pmatrix} x(i+n) \end{aligned} \tag{5.19}$$

We can now calculate the Bernstein coefficient with respect to x and we obtain a polynomial in $\tilde{x}$.

$$b_i^v(p)(\tilde{x}) = p(v(i,\tilde{x})) \quad \forall i \in \{0,1\}^{2n} \tag{5.20}$$

Observation 5.1. *The function $v(x,\tilde{x})$ is a linear function respect to $\tilde{x}$.*

This observation is easy to prove looking at the definition in Equation 5.19.

Observation 5.2. *The function $p(v(x,\tilde{x}))$ is a linear function respect to the couple of variables $(\tilde{x}(i), \tilde{x}(n+i))$.*

This observation can be proved using the third point of Property 5.1.

Suppose to consider only the joint i and to fix to 0 all the other values of ϵ, we have $\tilde{x} = (1, \ldots, \tilde{x}(i), \ldots, 1, 0, \ldots, \tilde{x}(n+i), \ldots, 0)$. Each Bernstein coefficient is defined by the following equation

$$b_j^v(p)(\tilde{x}) = p(v(j,\tilde{x})) \quad \forall j \in \{0,1\}^{2n} \tag{5.21}$$

and thanks to the Observation 5.2 we have that

$$\begin{aligned} b_j^v(p)(\tilde{x}) = & A_j\tilde{x}(i) + B_j\tilde{x}(n+i) + C_j \\ = & A_j\cos(\epsilon_i) + B_j\sin(\epsilon_i) + C_j \end{aligned} \tag{5.22}$$

with $A_j, B_j, C_j \in \mathbb{R}$.

We can use this last expression to compute the max-safety region with respect to the red region.

5.4.1 Half-space region

We consider the base case of a red region defined by a single constraint. Thus an entire half-space that define our red region. We can choose, for example, that the red

half-space is define by the following inequality

$$Y \leq K \quad K \in \mathbb{R} \tag{5.23}$$

Our starting point $P = T(\theta)$ is outside the red region so $P_y > K$. If we fix n values ϵ_i for each $i \in \{1, \ldots, n\}$, we know that the reachability region can be overapproximate using the Bernstein coefficients. We also know that $Y_{\min}$ is the minimum Bernstein coefficient of the polynomial p_y. So if we ensure that $Y_{\min} > K$ we are sure that our reachable region does not cross the red region.

We have to work with one variable ϵ_i per time and we need to search the maximum range such as each Bernstein coefficient does not exceed the K value.

In order to build a problem that has only one possible solution we can decide a priority between two different joints. We can choose to process all the joints, from the first to the last. A possible algorithm is Algorithm 5.3

Algorithm 5.3 Max-safety: half-space region

Input: K value that define the half-space bound.

Initialization: $\epsilon = (0, \ldots, 0)$
for $i = 1 : n$ **do**
  $\epsilon_i = 90°$
  for $j = 1 : 2^{2n}$ **do**
    find the maximum $\epsilon_i^{\max} \in [0, 45°]$ such as
    $A_j \cos(\epsilon_i^{\max}) + B_j \sin(\epsilon_i^{\max}) + C_j > K$ ▷ see Eq.(5.22)
    if $\epsilon_i^{\max} < \epsilon_i$ **then**
      $\epsilon_i = \epsilon_i^{\max}$
    end if
  end for
end for

Unfortunately this algorithm is highly dependent on the order and we should test all the possible configurations in order to maximize the volume of the reachable region. We illustrate this problem through an example.

Example 5.4.1. *Given the robot Racer*5-0.63 *with a starting position* $\theta = (90°, -90°, 0°, 0°, 0°, 0°)$ *we want to solve the max-safety problem and define the vector* ϵ *such as the reachability region doesn't cross the plane* $y = -500mm$. *We solve the max-safety problem processing the joints in two different orders. We select* $\pi_1 = \{1, 2, 3, 4, 5, 6\}$ *and* $\pi_2 = \{6, 5, 4, 3, 2, 1\}$. *In Figure 5.16 there is in red the plane and in blue the overapproximations of the maximum reachability regions. On the left we can see the obtained region with the order* π_1, *the maximum allowed angle is* $\epsilon = (45°, 7.29°, 0°, 0°, 0°, 0°)$. *On the right picture we plot the region with the order* π_2 *and we obtain* $\epsilon = (10.58°, 45°, 45°, 45°, 45°, 45°)$

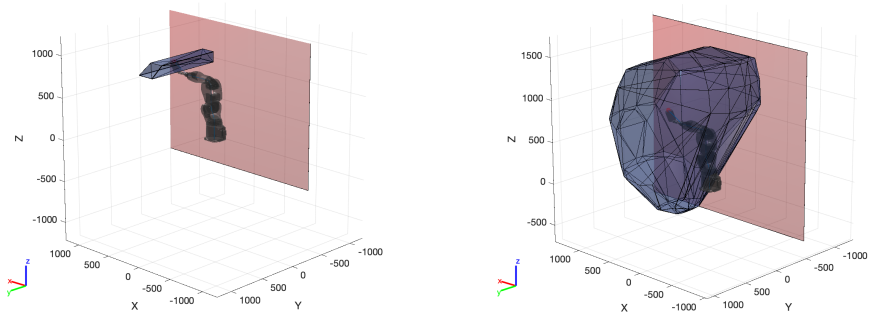


Figure 5.16: Two maximum reachable regions. On the left we select the order $\{1, 2, 3, 4, 5, 6\}$, on the right we use the reverse order $\{6, 5, 4, 3, 2, 1\}$

5.4.2 Polytope region

Lets consider a red zone $\mathcal{R}$ in a general form of a polytope defined with a set of linear inequalities.

$$\mathcal{R} := \{x \in \mathbb{R}^3 | A_i x \leq b_i \quad \forall i = 1, \ldots, m\} \tag{5.24}$$

with $A_i \in \mathbb{R}^3$ and $b_i \in \mathbb{R}$.

The method of Bernstein coefficients allowed to overapproximate the reachability region just with a convex region. When also the red region is a polytope, so a convex space, we can reduce the general case to the previous case of the half-space. Indeed is enough to conduct the research on the orthogonal direction to the minimum distance between the starting point and the red region.

5.5 Conclusion and future works

We reformulated the obstacle avoidance problem as a reachability problem considering the robot arm as a trigonometric dynamical system. We used a reachability technique based of Bernstein theory and we described in detail which are the steps to follow in order to adapt this technique with a kinematic chain robot. We focused on two problems: the safety problem to detect a possible collision and the max-safety problem to define a joint limitation in order to prevent the collision and guarantee the safeness of the reachable region.

In this chapter we collected all the basic ingredients to implement a reachability tool. A refinement of the implementation, especially for the max-safety problem, is needed in order to reduce the computational time and to ensure the applicability of the method on real case studies.

With this study we showed the feasibility of a more verification-oriented approach and we illustrated the versatility of Bernstein theory. In the next chapter we will continue to use this technique and we will adapt it on a completely different system.

6

Output set approximation for Feed Forward Neural Nets

Neural networks (NN) are increasingly used in the development of control systems in many autonomous applications such as robots, self-driving vehicles, and medical devices, thanks to their successful applications in fields such as image classification, natural language processing and speech recognition (see for example [68, 61] and references therein). Their ability of "learning" from data, used to describe appropriate behaviors and adapt to new situations, makes NN suitable to control problems in complex and changing environments. We call systems where NN are used to control a physical process Neural Net Control Systems (NNCS). However, assuring the safety and reliability of their usage, in particular under the impact of adversarial inputs or perturbations, is still challenging. Formal verification of neural networks has thus recently attracted much interest.

Robustness of neural networks can be theoretically verified using the tools developed in the context of abstract interpretation and programs verification to propagate uncertain input sets through the computation of the net. Nevertheless, these tools must face scalability issues when dealing with NN with an increasingly large number of layers and neurons. The verification problem becomes even more challenging for NNCS. Verifying NN components (without taking into account the effect of their decisions on the entire system behavior) is not sufficient to deduce the correctness and robustness of the closed-loop system, since NN accuracy measured in terms of a loss function may not reflect the closed-loop performance. A neural net that matches very well the training data may violate a desired closed-loop specification while another neural net with larger matching error can satisfy the specification. To achieve an acceptable compromise between accuracy and computation cost, it is thus of interest to be able to approximate the image of the function of the NN with given error bound.

In this work, we propose a method to do so, with view of applications to closed-loop verification of NNCS. Indeed the method can then be connected with an existing tool for continuous/hybrid systems (such as SpaceEx [45], CORA [4], Flow* [24]) to check closed-loop specification.

The NNCS verification problem but in particular NN verification problem has recently been widely investigated (see [73] for a survey). We can find a large variety

of methods and approaches like Mixed-Integer Linear Programs (MILPs) and global optimization [67, 35, 92], polytopes [48, 13], network conversion [56], linearization and function approximation [55, 118], Satisfiability Modulo Theory (SMT) [63, 64], Interval Arithmetic [114] and Set-based methods [12, 107].

The majority of them can handle only Rectified Linear Unit (ReLU) activation functions. The approach proposed in [56] can handle sigmoid activation functions only. The works in [55, 92] are more general and they use the Lipchitz theory to propagate the approximation. These approaches do not take advantage of the structures of the neural network and the types of activation functions. Some of these approaches can handle also sigmoid or hyperbolic tangent function but they have more refinement strategy on the ReLU case, e.g. ERAN tool [13]. The tool NNV [107] handles the same classes of activation functions of ERAN tool and uses also the monotonicity of some of them in the resolution of optimization problem for over-approximating the reachable sets.

The method we propose can handle most common classes of activation functions, namely ReLU, sigmoids, tanh. Our focus is on sigmoid and tanh activation functions, the ReLU case is extensively covered. Nevertheless the idea behind our approach is applicable to the majority of activation functions. Indeed it can be proved that these activation functions can be approximated by a rational function $r(\mathbf{x})$ (ratio of two polynomials) as accurately as desired [102]. Following this idea, we want to approximate each layer with a rational function and, assuming that the input lies inside a polytope, compute the image using the Bernstein expansion.

6.1 Problem definition

We consider a feed-forward neural network $N : \mathbb{R}^n \to \mathbb{R}^m$ with $L > 1$ layers. We indicate with n_l the number of neurons in the l^{th} layer, with $l = 1, \ldots, L$, and $n_1 = n$ and $n_L = m$. Given an input value $x \in \mathbb{R}^{n_{l-1}}$ of the l^{th} layer, we denote by h^l the function mapping x to the output of the layer:

$$h^l(x) = \sigma(W^l x + a^l) \quad l = 2, \ldots, L \tag{6.1}$$

where σ is the activation function, $W^l \in \mathbb{R}^{n_{l-1} \times n_l}$ and $a^l \in \mathbb{R}^{n_l}$. The output of the neural network is the composition of these functions:

$$N(x) = (h^L \circ h^{L-1} \cdots \circ h^2)(x), \qquad x \subseteq \mathbb{R}^n \tag{6.2}$$

Definition 6.1.1. (Reachability problem for NN) *Given an input set $X \subseteq \mathbb{R}^n$ of the NN, we want to over-approximate the image $N(X) = \{N(x) \mid x \in X\}$ within a desired over-approximation error bound ε.*

We address the reachability problem for *polytopes*, a class of convex sets commonly used in the reachability problem.

Definition 6.1.2. (Polytope) *A* polytope $P \subset \mathbb{R}^n$ *is a bounded subset of $\mathbb{R}^n$ such that there is a finite set $H = \{h_1, \ldots, h_m\}$ of half spaces whose intersection is P, i.e.:*

$$Q = \bigcap_{i=1}^{m} h_i, \tag{6.3}$$

where an half-space is a set $h = \{x \mid Ax \leq b\}$ *with* $d \in \mathbb{R}^n \setminus \{0\}$ *and* $c \in \mathbb{R}$.

In literature, a polytope is often said a $\mathcal{H}$-polytope if it is represented as a set of half-space. However, a polytope can also be seen as the convex hull of a finite set of points $V = \{v_1, \ldots, v_p\}$. A polytope represented b y its vertices is called a $\mathcal{V}$-polytope. We also list two important properties.

Property 6.1. (Parallelotope decomposition) *There exists a finite set of parallelotopes* $\{P_1, \ldots, P_k\}$ *such that* $X = \cap_{i=1}^{k} P_i$, *where* parallelotopes *is the n-dimensional generalization of a parallelogram, i.e. a symmetric convex polytope whose opposite facets are parallel.*

Property 6.2. (Simplex decomposition) *There exists a finite set of siplices* $\{S_1, \ldots, S_t\}$ *such that* $X = \cup_{i=1}^{k} S_i$, *with* $S_i \cap S_j = \emptyset$ *for* $i \neq j$.

Given a neural network N a polytope X as the input set, we want to approximate layer by layer the non-linear function h^l and we can bound the output of the approximated function using the Bernstein theory as we did in Chapter 5 for the arm robot application.

Lets focus on the function in Equation (6.1) for a generic layer l with m neurons and n input for each neuron, i.e. $h^l : \mathbb{R}^n \to \mathbb{R}^m$. For simplicity of notation, in the sequel we drop the superscript indicating the indices of the layer.

$$y_i = h(x) = \sigma(W_i x + a_i) \tag{6.4}$$

with $W \in \mathbb{R}^{n \times m}$, $a \in \mathbb{R}^m$ and we denote with $x \in \mathbb{R}^n$ the input values and with $y \in \mathbb{R}^m$ the output values.

In order to use Bernstein we need to approximate the non-linear activation function with a polynomial function or a rational one. Thanks to the result in [102] we can use a rational approximation.

6.2 Bernstein approximation with a Rational Function

We already know from Chapter 2.2.3 that the convex hull property does not hold for rational functions but we can just use the range enclosing property. We recall this property.

Definition 6.2.1. (Range enclosing property) *Considering a rational function* $f = \frac{p}{q}$ *where* p, q *are polynomials. We compute the Bernstein coefficients* $b^s_{\mathbf{i}}(p)$ *and* $b^s_{\mathbf{i}}(q)$ *over a domain* X *which can be a box or a standard simplex, with* s *greater than the degrees of* p *and* q. *Assuming that all Bernstein coefficients* $b^s_{\mathbf{i}}(q)$ *have the same sign and are non-zero (which implies that* $q(\mathbf{x}) \neq 0$, *for all* $\mathbf{x} \in X$*), then the range enclosing property holds*

$$\min_{\mathbf{i} \leq s} \frac{b^s_{\mathbf{i}}(p)}{b^s_{\mathbf{i}}(q)} \leq f(\mathbf{x}) \leq \max_{\mathbf{i} \leq s} \frac{b^s_{\mathbf{i}}(p)}{b^s_{\mathbf{i}}(q)} \tag{6.5}$$

We can apply the range enclosing property to a generic polytope by splitting it into siplices with Property 6.2 or we can decompose it into parallelotopes using Property 6.1 and transform the standard box into the parallelotopes with an affine transformation (a generalization of Equation (5.17)).

To apply the range enclosing property we need also to guarantee that the Bernstein coefficients of the denominator have the same sign. We already discuss in Chapter 2.2.3 a possible way to guarantee this property. Lets consider a particular activation function to understand how we can apply it in practice.

We select the hyperbolic tangent as representative of an activation function. The hyperbolic tangent is defined as

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6.6}$$

It is known that tanh can be defined also with the Lambert's continued fraction defined as

$$\tanh(x) = \cfrac{x}{1 + \cfrac{x^2}{3 + \cfrac{x^2}{5 + \ldots}}} \tag{6.7}$$

We can approximate the hyperbolic tangent function taking Equation (6.7) up to a finite number of fraction and we will call $r_d(x)$ the rational function with d fractions. In Figure 6.1 we plot the approximation in the domain $[-10, 10]$.

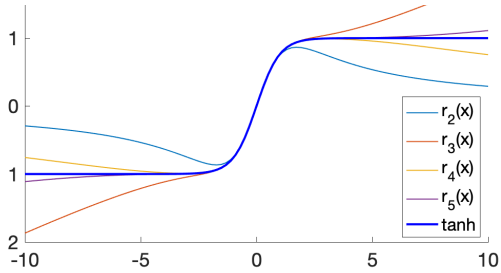


Figure 6.1: Approximation of tanh with Lambert's fraction.

Given a confidence interval $[lb, ub] \subset \mathbb{R}$ we approximate the Equation (6.4) with

$$y_i \approx \begin{cases} -1 & x < lb \\ r_d(W_i x + a_i) & lb \leq x \leq ub \\ 1 & x > ub \end{cases} \tag{6.8}$$

We can observe that if d is odd we just need to bound/intersect the output range with $[-1, 1]$.

It is easy to observe that the denominator of $r_d(x)$ is a positive even polynomial function with degree $\lfloor \frac{d}{2} \rfloor$ and with only positive coefficients. It is also immediate to

prove that this polynomial has a minimum at $x = 0$ with value $\prod_{i=0}^{d-1} 2i+1$. Theorem 2.1 suggests us that the Bernstein coefficients of the denominator function are positive if we select the couple $(0, b^s_{\boldsymbol{\alpha}}(p))$ in our control points. Moreover, by construction, the extreme points of a domain are always control points. For this reason we decide to split our domain with $x \geq 0$ and $x \leq 0$ in order to guarantee that all the Bernstein coefficients of the denominator are positive.

By changing the activation function, thus the rational approximation, we need to perform the same analysis before applying the Bernstein approximation.

To conclude we sketch in Algorithm 6.4 the pseudocode of the complete procedure to overapproximate the image of a complete NN with an input set X. We consider a neural net where for each layer $l = 1, \ldots, L$, W^l and a^l denote the matrix and bias of the layer, and $r^l_d = \frac{p^l}{q^l}$ is the rational approximation of the activation tanh function, with d odd. Hence for layer l we have $p^l = p(W^l\mathbf{x} + a^l)$; $q^l = q(W^l\mathbf{x} + a^l)$.

In line 4 we split the input domain to guarantee a positive certificate. In line 5 we are using Property 6.2 to apply in line 7 the range enclosing property.

Algorithm 6.4 Approximation of a NN with tanh using rational approximations.

1: $X_{curr} = X$
2: **for** $l = 1, \ldots, L$ **do**
3: **for** $k = 1 : n_l$ **do**
4: calculate $X^+ = X_{curr} \cap \{W^l_k\mathbf{x} + a^l_k \geq 0\}$ and $X^- = X_{curr} \cap \{W^l_k\mathbf{x} + a^l_k \leq 0\}$
5: split both X^+ and X^- into t and s simplices $S_1, \ldots, S_t$ and $P_1, \ldots, P_s$
6: compute $b_{\mathbf{i}}(X^+) = \left\{\frac{b_{\mathbf{i}}(S_j, p^l_k)}{b_{\mathbf{i}}(S_j, q^l_k)}\right\}_{j=1,\ldots,t}$ and $b_{\mathbf{i}}(X^-) = \left\{\frac{b_{\mathbf{i}}(S_j, p^l_k)}{b_{\mathbf{i}}(S_j, q^l_k)}\right\}_{j=1,\ldots,s}$
7: define $\underline{b} = \min_{\mathbf{i}}\{b_{\mathbf{i}}(X^+), b_{\mathbf{i}}(X^-)\}$ and $\overline{b} = \max_{\mathbf{i}}\{b_{\mathbf{i}}(X^+), b_{\mathbf{i}}(X^-)\}$
8: $\underline{y}_k = \max(\min(\underline{b}, 1) - 1)$ and $\overline{y}_k = \max(\min(\overline{b}, 1) - 1)$
9: **end for**
10: $X_{curr} = \prod_{i=1}^{n} [\underline{y}_i, \overline{y}_i]$
11: **end for**

In this case Property 6.1 it is more complex to be used. Indeed we need to introduce the affine transformation v for each parallelotope and split each one with respect to $v(W^l_k\mathbf{x} + a^l_k) \geq 0$ and $v(W^l_k\mathbf{x} + a^l_k) \leq 0$. There is no guarantee that the splitting domain is again a parallelotope and this can cause an additional splitting. It is complex also to change the output direction and we need to modify the splitting hyperplane also according also to the chosen direction. The Algorithm 6.4 is already heavy and we just bound the output of every layer with an hyperbox. In particular the splitting phase in line 5 can explode increasing the dimension, i.e. the number of neurons per layer.

As proposed in [13], one possibility to simplify the computational effort is to duplicate the neurons on each layer and analyze separately the affine computation $(Wx + a)$ and the activation function. If we keep an interval approximation we do not need to apply Bernstein but we can exploit the monotonicity of the activation function.

6.2.1 Exploiting monotonicity

We consider again the computation of a single layer in (6.1). We can split this computation in two steps

$$\mathbf{y} = W\mathbf{x} + a \tag{6.9}$$

$$\mathbf{z} = \sigma(\mathbf{y}) \tag{6.10}$$

Equation (6.9). The functions in (6.9) are m linear functions with n variables. The sharpness property allows to quickly compute the Bernstein coefficients. We have to translate each $\mathbf{x} \in X$ in the unitary cube $\mathbf{u} \in [0,1]^n$. We define $a^* := \underline{\mathbf{x}} + a$ and $W^* = (W^l_{i,j}(\overline{x}_j - \underline{x}_j))_{i,j}$. For the sharpness property we can compute the Bernstein coefficients as $b_{\mathbf{i}}(W\mathbf{x}+a, X) = W^*\mathbf{i} + a^*$ with multi-index $\mathbf{i} \in \{0,1\}^n$. We need only the minimum and the maximum $b_{\mathbf{i}}$, so we can simplify the search by looking at the positive and negative values of W^*. In particular if we want to compute the bound for y_j, $j = 1, \ldots, m$, we consider the row W^*_j and obtain $\sum_{i:W^*_{i,j}<0} W^*_{i,j} + a^*_j \leq y_j \leq \sum_{i:W^*_{i,j}>0} W^*_{i,j} + a^*_j$. We indicate this interval with $[\underline{y}_j, \overline{y}_j]$.

Equation (6.10). The activation function is a monotone function over the interval $[\underline{y}_j, \overline{y}_j]$ and we can easily compute the image interval approximation by applying the specific activation function. So we have $\mathbf{z} \in \prod_{j=1}^n [\sigma(\underline{y}_j), \sigma(\overline{y}_j)]$. We mention that monotonicity is also exploited in [116] in the formulation of optimization problem for bounding the output variables. The structure of this algorithm is reported in Algorithm 6.5.

Algorithm 6.5 Box-approximation directly with the exact activation function.

Input: $X = \prod_{i=1}^n [\underline{x}_i, \overline{x}_i]$ domain; W^l, a^l, $l = 1, \ldots, L$, matrix and bias of the neural network for each layer; σ activation function;
$X_{curr} = X$
for $l = 1, \ldots, L$ **do**
  calculate $W^* = (W^l_{i,j}(\overline{x}_j - \underline{x}_j))$, $a^* = a^l + \underline{x}$
  for $k = 1 : n_l$ **do**
    $\underline{y}_k = \sum_{i:W^*_{i,k}<0} W^*_{i,k} + a^*_k$ and $\overline{y}_k = \sum_{i:W^*_{i,k}>0} W^*_{i,k} + a^*_k$ ▷ exploiting Equation (6.9)
    $[\underline{z}_i, \overline{z}_i] = [\sigma(\underline{y}_k), \sigma(\overline{y}_k)]$ ▷ exploiting Equation (6.10)
  end for
  $X_{curr} = \prod_{i=1}^n [\underline{z}_i, \overline{z}_i]$
end for

Experimental result

Algorithm 6.5 is faster, scalable and is applicable for tanh, sigmoid, ReLU or other monotonic activation functions. This algorithm can be used as a quick approximation to decide if a more accurate approximation is needed for property verification.

To illustrate the potentiality of this algorithm, we use a Simulink® model of a single link robotic arm, provided by Mathworks[1]. The aim is to control the trajectory

of this system while the value of the input reference signal $r(\cdot)$ randomly changes in $[-0.5, 0.5]$ every 10 seconds. We use a nominal discrete-time feedback PID controller to generate desired trajectories as data for training a neural net of the following form $u_k = NN(r_k, \ldots, r_{k-3}, y_k, \ldots, y_{k-3}, u_{k-1}, u_{k-2})$ where y is the output, u is the control input updated at discrete time points, r is the reference that the output should track. We train two different neural nets with 2 hidden layers with 30 neurons each. We call the first neural network NN_{bad} since it has a diverging behavior as shown in Figure 6.2 for the constant reference with value -0.2. The second is called NN_{ok} since it has a correct behavior shown in Figure 6.3.

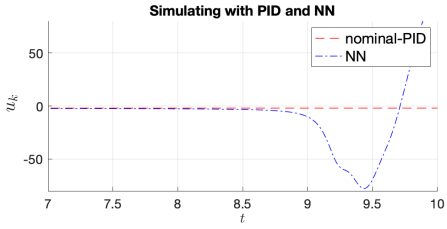


Figure 6.2: Comparison of the control output for NN_{bad} and a standard PID controller.

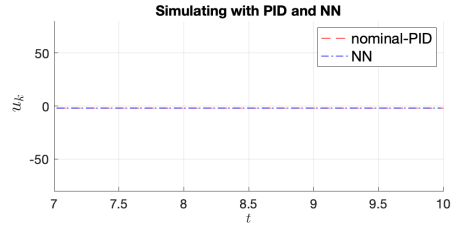


Figure 6.3: Comparison of the control output for NN_{ok} and a standard PID controller

We define a noise factor $\eta = |r| \cdot p$ where p has three values $\{0.01\%, 0.1\%, 0.5\%\}$. We run Algorithm 6.5 where the first 4 inputs $r_k, \ldots, r_{k-3}$ take constant value -0.2, the next 4 inputs have the range $[-0.2 - \eta, -0.2 + \eta]$ and the last two have the range $u = 10y$. We obtain the approximations in Table 6.1.

Table 6.1: Interval approximation of NN_{bad} and NN_{ok} using Algorithm 6.5 and adding the noise factor η.

	$p = 0.01\%$	$p = 0.1\%$	$p = 0.5\%$
NN_{bad}	[-2.344, -1.684]	[-5.342, 1.313]	[-18.673, 14.600]
NN_{ok}	[-2.133, -1.862]	[-3.368, -0.627]	[-8.850, 4.858]

We can see that the output interval of NN_{bad} is 2.4 times the size of the output interval of NN_{ok}, and this result is coherent with the behavior of the two nets.

6.3 Bernstein approximation with a Polynomial Function

We have understood the potentialities and the limitations of the rational approach. In order to improve the current strategy, at least for sigmoid and tanh functions, we

[1] `https://ch.mathworks.com/help/deeplearning/ug/design-model-reference-neural-controller-in-simulink.html`

consider a polynomial approximation. W.l.o.g. we continue to consider the hyperbolic tangent. Indeed we can notice that the sigmoid function $\sigma(x) = \tanh(\frac{x}{2}) + 1$ and vice versa $\tanh(x) = 2\sigma(2x) - 1$. If we select a polynomial approximation $p(x) \approx \tanh(x)$ than $q(x) = p(\frac{x}{2}) + 1$ is a polynomial approximation for $\sigma(x)$.

As in the rational case, we approximate the hyperbolic tangent with a piecewise function in a confidence interval $[lb, ub] \subset \mathbb{R}$.

$$\tanh(x) \approx \begin{cases} -1 & x < lb \\ p(x) & lb \leq x \leq ub \\ 1 & x > ub \end{cases} \tag{6.11}$$

We consider three types of polynomial approximation: Taylor, Chebyshev and minmax approximation.

The first way to approximate a function with a polynomial expression is by taking the Taylor expansion up to a certain degree. Given a function $f : \mathbb{R} \to \mathbb{R}$ we can write the Taylor expansion in x_0 as

$$f(x_0) + f^{(1)}(x_0)(x - x_0) + \frac{f^{(2)}(x_0)}{2!}(x - x_0)^2 + \dots \tag{6.12}$$

with $f^{(i)}$ is the i-th derivative of f. For the hyperbolic tangent we use the Taylor expansion in $x_0 = 0$ and we have

$$\tanh(x) \approx x - \frac{x}{3} + \frac{2x^5}{15} + \dots \tag{6.13}$$

In Figure 6.4 we plot different approximations with a saturation to $[-1, 1]$ if we exceed this interval.

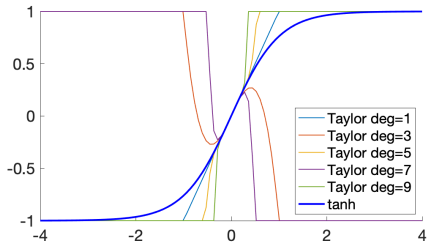


Figure 6.4: Approximation of hyperbolic tangent with Taylor expansion around $x_0 = 0$ from degree 1 up to degree 9.

This approximation is really rough and it approximates the function only locally.

In [111] they suggest to use the Chebyshev expansion to better approximate sigmoid functions, and so also the hyperbolic tangents. They define the Chebyshev expansion as the integration of the Bernstein expression as follows:

$$C_{p,q}(x) = \left(\frac{1+x}{2}\right)^{q+1} \sum_{i=0}^{p} \binom{i+q}{i} \left(\frac{1-x}{2}\right)^{i} \tag{6.14}$$

Vlček obtains approximation of $\sigma(8x)$ with the expression $C_{11,11}(x)$. We plot in Figure 6.5 the modified polynomial $2 \cdot C_{11,11}(x/4) - 1$ in comparison with the hyperbolic tangent.

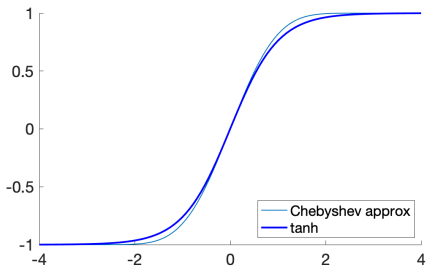


Figure 6.5: Chebyshev approximation of the hyperbolic tangent.

This is a better approximation in compare with the Taylor one, but we are working with a polynomial of degree 23 and we still have a maximum error of 0.06 that is the 3% of the output range.

A third possibility is to consider a minmax approximation. Given a desire degree we can sample the hyperbolic tangent in the confident interval and we can tune the coefficients of the polynomial function in order to minimize the maximum absolute error. For example we can use the *polyfit* function available in MATLAB®to obtain the polynomial functions in Figure 6.6. With this approximation we create a polynomial function of degree 9 and with a maximum error of 0.0165.

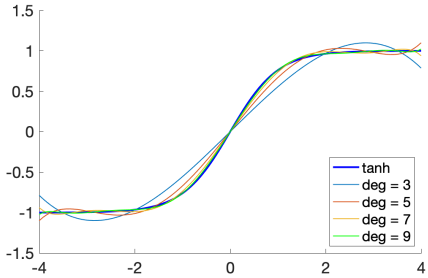


Figure 6.6: Minmax polynomial approximation of the hyperbolic tangent using the *polyfit* tool in MATLAB®.

Minmax approximation allows to reduce the total error and we also decrease the polynomial degree. We can now use it to perform a Bernstein overapproximation. We select the polynomial function $\pi(x)$ with degree $d = 9$ and we can observe, similarly to the rational case, that if we select the right degree ($\frac{d-1}{2}$ even) it is enough to bound the output in $[-1, 1]$.

Instead of implementing a new procedure to compute the Bernstein approximation, with the polynomial form we can compute the image approximation by calling some

of the already implemented functions inside the SAPO tool [33]. To use the SAPO tool we need to transform a polytope into its bundle representation.

Lets consider a polytope X as an input set defined by a list of inequalities $Ax \geq b$. Each line of the matrix A correspond to a direction and b are the corresponding offsets. We know from Property 6.1 that we can split X into a set of parallelotopes. We call this equivalent form a *bundle representation*. Lets assume that the polytope X has n_{dir} possible directions, n variables and we decompose it into n_{parall} parallelotopes. In this representation we specify a matrix $L \in \mathbb{R}^{n_{dir} \times n}$ of directions and two vectors $\underline{c} \in \mathbb{R}^{n_{dir}}$, $\overline{c} \in \mathbb{R}^{n_{dir}}$ such as $Lx \leq \overline{c}$ and $-Lx \leq \underline{c}$. Then we define a matrix $T \in \{1, \ldots, n_{dir}\}^{n_{parall} \times n}$ where we group the directions of L to form the different parallelotopes. The polytope $X =< L, T, \underline{c}, \overline{c} >$ is expressed in the bundle representation.

In general the number of directions are less or equal the number of the initial inequalities. For example if we consider the unitary box in 2D defined as $[0,1]^2 = \{x \in \mathbb{R}^2 | Ax \geq b\}$ with

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 0 \\ -1 \\ 0 \\ -1 \end{pmatrix}$$

the associated bundle representation has

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 2 \end{pmatrix} \qquad \underline{c} = \begin{pmatrix} 0 & 0 \end{pmatrix} \qquad \overline{c} = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

The final algorithm is reported in Algorithm 6.6. The steps that are done inside the third for-loop are already implemented in SAPO.

Algorithm 6.6 Approximation of a NN with tanh using polynomial approximations.

1: $X_{curr} = X$ defined as $X =< L, T, \underline{c}, \overline{c} >$
2: **for** $l = 1, \ldots, L$ **do**
3: $\quad \underline{c} = -\infty$, $\overline{c} = -\infty$
4: $\quad$ **for** $i = 1 : n_{parall}$ **do**
5: $\quad\quad$ **for** $j = 1 : n_{dir}$ **do**
6: $\quad\quad\quad$ P is the i-th parallelotope of X_{curr}
7: $\quad\quad\quad$ define $v : [0,1]^{n_{l-1}} \to P$
8: $\quad\quad\quad$ define $f = \pi(W * v(x) + a)$ with $f : [0,1]^{n_{l-1}} \mathbb{R}^{n_l}$
9: $\quad\quad\quad$ compute the coefficients b_i for $L_j \cdot f(x)$
10: $\quad\quad\quad$ update $\underline{c}_j = \max(\underline{c}_j, \max(b_i))$, $\overline{c}_j = \max(\overline{c}_j, -\min(b_i))$
11: $\quad\quad$ **end for**
12: $\quad$ **end for**
13: $\quad$ $X_{curr} =< L, T, \underline{c}, \overline{c} > \cap [-1,1]^{n_l}$
14: **end for**

With the polynomial form we obtain a thin approximation because we can change both the input and the output directions. In addiction, we reduce the computational

time avoiding the split into simplices. The computation of the Bernstein coefficients in Line 9 is still a tough point and it is hardly dependent on the number of neurons in the previous layer.

6.4 Conclusion and future works

The increasing use of NN in many autonomous applications leads to an interest of developing tools for NN formal verification. The verification problem becomes even more difficult for NNCS where the performance of the net may not reflect the behavior of the closed loop.

A major challenge in verifying NNCS is to be able to approximate the image of a NN within a given error bound. We describe a method using rational approximations for activation functions and taking advantage of the Bernstein expansion. We sketch this approach in an abstract algorithm. Moreover, by monotonicity of the activation functions, we propose another, faster and more scalable, algorithm. This algorithm can be used as a quick approximation to decide if a more accurate approximation is needed for property verification. The two proposed algorithms use box approximations, in the last section we discussed a possible improvement using polynomial approximation when we consider hyberbolic tangent or sigmoid activation functions. The polynomial approximation enables the possibility to obtain approximated NN images more accurately with more complex sets, such as polytopes.

The proposed methods have just a prototype implementation to validate the idea but we did not reach the maturity to test them against some benchmark case study. Nonetheless it is evident that the computation of Bernstein coefficients is a stopping point when we consider huge networks, because we are handling a function over n_{l-1} variable, that are all the neurons at the previous layer (or at the input layer). A possibility to build a scalable tool is to analyze a fixed subset of neurons at a time, as it is proposed in the ERAN tool for ReLU activation functions. We are currently working to implement the designed strategy using the ERAN tool as the main framework.

Conclusions

The fourth industrial revolution and the call for smart functionalities and intelligent products is pervasive in any type of business. Food service sector is not an exception. Electrolux Professional, as a leader in this sector, is continuously researching for new techniques to improve the design process or directly the final products to help professional kitchens to become efficient and manageable.

In this technological scenario, where computational and dynamical/physical capabilities as deeply intertwined inside any device and appliance, the emerging type of systems are the Cyber Pysical Systems (CPS). This thesis, funded by Electrolux Professional, starts the analysis of CPS in food service sector focusing on optimization problems and reachability analyses. We present four different analyses with a direct impact on the professional sector.

The first project, related to the scheduling of a professional oven, has the most concrete outcome and it answers to the demanding needs for productivity and saving in food catering, canteens and food retail services. This functionality is already included in the controller of the professional ovens and it is able to sort a list of cooking recipes in order to minimize the total energy consumption of the oven. The energy approximation that we design allows to solve the scheduling problem over multistage systems as a standard Traveling Salesman Problem (TSP).

It is a different story for the thawing multistage system analyzed in Chapter 4. The challenge is the selection of the complex parameters for this multistage system in order to balance the productivity and quality of the global process using a new thawing technology: the air impingement thawing. We propose two approaches. The availability of computationally efficient models enables a multi-objective analysis using a genetic algorithm. This allows to obtain a set of optimal configurations to be further tested in a prototype for a food quality analysis. Unfortunately, with this analysis we were not able to consider the safeness of the multistage system as a constraint. For this reason, we proposed a second approach where we exploit the structure of the multistage system to define a hybrid automaton and we reformulate the optimization problem as a reachability analysis.

With this second study we build a bridge between the optimization field and the tools used for verification analysis. We open here the second topic of this thesis and the next two projects deal mainly with reachability analysis keeping an eye on optimization problems. We focus on the verification of advanced tools that are increasingly presents in professional kitchens, i.e. collaborative robots and neural networks by selecting a specific reachability tool based on the Bernstein approximation for polynomial functions.

The Bernstein technique used in for a collaborative robot not only allows to avoid an obstacle but can guarantee that all the possible movements of the robot will not collide with the given obstacle. The search of this guarantee is and optimization

problem and the constraint itself is the maximum angular range for each joint of the robot.

Finally, the last analysis touches another important trend of these years: Neural Networks (NN). We focus on the verification of neural net control systems (NNCS), i.e. systems where the NN are used to control the physical process. Neural network are model approximation tools and, in a broad sense, can be classify as optimization tools. In this project we are not using NN as a tool, but the aim is analyzing this tool and be able to estimate and bound the behavior of the neural net. We propose a method that can handle most common classes of activation functions. The idea is to process the network layer by layer and approximate the layer with rational or polynomial function in order to bound the output image using the Bernstein expansion.

Future works have been already discussed at the end of each chapter and all these four analyses open a possible field of research. In particular the latter topic, as it is emerging from the most recent literature in the field, represents a fundamental tool to deal with NNCS. Moreover we want to highlight that the parameter synthesis study conducted in Chapter 4 can be further enlarged by changing the virtual model with a data-driven one. Indeed recent studies are combining combinatorial optimization tools directly in the training phase of a NN [21, 74] to solve parameters optimizations, and this would be an additional step to gain the best of both optimization and machine learning fields. We strongly believe that the challenge for the future, that will be a key point for handling complex systems and improve our ability on controlling and modeling such systems, is the cross collaboration of different fields as optimization and verification but also thermodynamics, control theory and machine learning.

List of publications and patents

[BPTM21] Arianna Bozzato, Eleonora Pippia, Emidio Tiberi, and Lara Manzocco. Air impingement to reduce thawing time of chicken fingers for food service. *Journal of Food Processing and Preservation*, 2021. Manuscript submitted for publication.

[LPR20] Giuseppe Lancia, Eleonora Pippia, and Franca Rinaldi. Using integer programming to search for counterexamples: A case study. In *Mathematical Optimization Theory and Operations Research (MOTOR 2020)*, pages 69–84, Cham, 2020. Springer International Publishing. `http://doi.org/10.1007/978-3-030-49988-4_5`.

[PBT+19] Eleonora Pippia, Arianna Bozzato, Emidio Tiberi, Riccardo Furlanetto, and Alberto Policriti. Optimization and multistage systems. the thawing case. In *Artificial Intelligence and fOrmal VERification, Logic, Automata, and sYnthesis OVERLAY@AI*IA*, 2019. `http://ceur-ws.org/Vol-2509/paper16.pdf`.

[PBT20] Eleonora Pippia, Arianna Bozzato, and Emidio Tiberi. Multi-objective optimization of multistage jet impingement thawing processes. *Computers & Industrial Engineering*, 149:106771, 2020. `https://doi.org/10.1016/j.cie.2020.106771`.

[PDP20] Eleonora Pippia, Thao Dang, and Alberto Policriti. Image approximation for feed forward neural nets, 2020. `https://sites.google.com/view/vnn20/program?authuser=0#h.vh8ckqz9khgr`.

[SFPT19] Michele Simonato, Riccardo Furlanetto, Eleonora Pippia, and Emidio Tiberi. Method for operating a cooking oven. 2019. Patent:EP3702673A1.

Bibliography

[1] C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer, Cham, 2018. http://doi.org/10.1007/978-3-319-94463-0.

[2] T. Akhtar and C.A. Shoemaker. Efficient multi-objective optimization through population-based parallel surrogate search. *ArXiv*, 2019.

[3] S. Ali, W. Zhang, N. Rajput, M. A. Khan, C. Li, and G. H. Zhou. Effect of multiple freeze–thaw cycles on the quality of chicken breast meat. *Food Chemistry*, 173:808–814, 2015.

[4] M. Althoff. An introduction to cora 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015. https://easychair.org/publications/open/xMm.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, and et al. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–35, 1995.

[6] R. Alur, N. Courcoubetis, C, T. A. Henzinger, and Pei-Hdin Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems*, 736:209–229, 1993.

[7] R. Alur, A. Trivedi, and D. Wojtczak. Optimal scheduling for constant-rate multi-mode systems. *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, pages 75–84, 2012.

[8] A. Azadeh, S. Elahi, M. Hosseinabadi Farahani, and B. Nasirian. A genetic algorithm-taguchi based approach to inventory routing problem of a single perishable product with transshipment. *Computers & Industrial Engineering*, 104:124–133, 2017.

[9] C. J. Backi. Methods for (industrial) thawing of fish blocks: A review. *Journal of Food Process Engineering*, 41:1–11, 2017.

[10] M. Badescu and C. Mavroidis. New performance indices and workspace analysis of reconfigurable hyper-redundant robotic arms. *The International Journal of Robotics Research*, 23(6):643–659, 2004. https://doi.org/10.1177/0278364904044406.

[11] R. Baheti and H. Gill. Cyber-physical systems. In *The Impact of Control Technology*, pages 161–166, 2011.

[12] S. Bak, H.-D. Tran, K. Hobbs, and T. T. Johnson. Improved geometric path enumeration for verifying relu neural networks. In *Computer Aided Verification*, pages 66–96, Cham, 2020. Springer International Publishing. https://doi.org/10.1007/978-3-030-53288-8_4.

[13] M. Balunovic, M. Baader, G. Singh, T. Gehr, and M. Vechev. Certifying geometric robustness of neural networks. In *Advances in Neural Information Processing Systems 32*, pages 15313–15323. Curran Associates, Inc., 2019. http://papers.nips.cc/paper/9666-certifying-geometric-robustness-of-neural-networks.pdf.

[14] A. Becker, A. Boulaaba, S. Pingen, C. Krischek, and G. Klein. Low temperature cooking of pork meat — physicochemical and sensory aspects. *Meat Science*, 118:82–88, 2016. https://doi.org/10.1016/j.meatsci.2016.03.026.

[15] C. Belta, L. C. G. J. M. Habets, and R. V. Kumar. Control of multi-affine systems on rectangles with applications to hybrid biomolecular networks. In *41st IEEE Conference on Decision and Control*, volume 1, pages 534–539, 2002.

[16] A. Bemporad and N. Giorgetti. A logic-based hybrid solver for optimal control of hybrid systems. *Conference on Decision and Control*, 2003.

[17] B. Bittner, M. Bozzano, A. Cimatti, M. Gario, and A. Griggio. Towards pareto-optimal parameter synthesis for monotonie cost functions. *2014 Formal Methods in Computer-Aided Design (FMCAD)*, pages 23–30, 2014.

[18] A. Blum, J. Hopcroft, and R. Kannan. *Foundations of Data Science*. Cambridge University Press, 2020. http://doi.org/10.1017/9781108755528.

[19] O. Bohigas, M. Manubens, and L. Ros. A complete method for workspace boundary determination on general structure manipulators. *Robotics, IEEE Transactions on*, 28:993–1006, 10 2012. https://doi.org/10.1109/TRO.2012.2196311.

[20] J. A. Bondy and U. S. R. Murty. *Graph Theory whith Applications*. Springer, 1976.

[21] A. Borghesi, G. Tagliavini, M. Lombardi, L. Benini, and M. Milano. Combining learning and optimization for transprecision computing. *Proceedings of the 17th ACM International Conference on Computing Frontiers*, 2020. http://dx.doi.org/10.1145/3387902.3392615.

[22] CAC/RCP 39. *Code of Hygienic Practice for Precooked and Cooked Foods in Mass Catering*, 1993. Codex Alimentarius.

[23] Alberto Casagrande. *Hybrid System: A First-Order Approach to Verification and Approximation Techniques*. PhD thesis, Università degli Studi di Udine, 2006. pp. 23-34.

[24] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013. https://doi.org/10.1007/978-3-642-39799-8_18.

[25] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. Parameter synthesis with ic3. *2013 Formal Methods in Computer-Aided Design, FMCAD 2013*, pages 165–168, 2013.

[26] C. A. C. Coello. *Evolutionary Optimization*. Springer, Boston, Massachusetts, 2003.

[27] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. https://doi.org/10.1007/BF02551274.

[28] X. Dai, X. Yuan, and Z. Zhang. A self-adaptive multi-objective harmony search algorithm based on harmony memory variance. *Applied Soft Computing*, 35:541–557, 2015.

[29] B. Danaei, N. Karbasizadeh, and M. Tale Masouleh. A general approach on collision-free workspace determination via triangle-to-triangle intersection test. *Robotics and Computer-Integrated Manufacturing*, 44:230–241, 2017. https://doi.org/10.1016/j.rcim.2016.08.013.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[31] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 825–830. IEEE, 2002.

[32] Tan Do. *Generalised Bernstein Operators on Polytopes*. PhD thesis, University of Auckland, 05 2011.

[33] T. Dreossi. Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems. *CoRR*, abs/1607.02200, 2016. http://arxiv.org/abs/1607.02200.

[34] T. Dreossi. Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems. *Hybrid Systems: Computation and Control (HSCC 2017)*, 2017.

[35] S. Dutta, X. Chen, S. Jha, S. Sankaranarayanan, and A. Tiwari. Sherlock - a tool for verification of neural network feedback systems: Demo abstract. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, pages 262–263, 2019. https://doi.org/10.1145/3302504.3313351.

[36] F. Erdogdu and R. P. Singh. Mathematical modeling of air-impingement cooling of finite slab shaped objects and effect of spatial variation of heat transfer coefficient. *Journal of Food Engineering*, 71:287–294, 2005.

[37] B. Ersoy, E. Aksan, and A. Özeren. The effect of thawing methods on the quality of eels (anguilla anguilla). *Food Chemistry*, 111:377–380, 2008.

[38] Andrè Étienne and Soulat Romain. *The Inverse Method*. Wiley, 2013.

[39] J. Evans, S. Russell, and S. James. Chilling of recipe dish meals to meet cook—chill guidelines. *International Journal of Refrigeration*, 19(2):79–86, 1996.

[40] R. Evins. A review of computational optimisation methods applied to sustainable building design. *Renewables and Sustainable Energy Reviews*, 22:230–245, 2013.

[41] G. Farin. Triangular bernstein-bézier patches. *Computer Aided Geometric Design*, 3(2):83–127, 1986.

[42] A. Fenucci, M. Indri, and F. Romanelli. A real time distributed approach to collision avoidance for industrial manipulators. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, 2014.

[43] R. Finotello, T. Grasso, G. Rossi, and A. Terribile. Computation of kinetostatic performances of robot manipulators with polytopes. In *1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 4, pages 3241–3246, 1998.

[44] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19 – 27, 2003. https://doi.org/10.1016/S0167-8396(03)00002-5.

[45] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification*, pages 379–395. Springer Berlin Heidelberg, 2011. https://doi.org/10.1007/978-3-642-22110-1_30.

[46] Riccardo Furlanetto. *Algoritmi di previsione e di controllo per processi di cottura, raffreddamento e congelamento*. PhD thesis, Università degli Studi di Trieste, 2007.

[47] Juergen Garloff. Convergent bounds for the range of multivariate polynomials. In *Interval Mathematics 1985*, pages 37–56, 01 1985.

[48] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2018. https://doi.org/10.1109/SP.2018.00058.

[49] S. M. Grigorescu, B. Trasnea, T. T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *CoRR*, abs/1910.07738, 2019. http://arxiv.org/abs/1910.07738.

[50] Y. Guan, K. Yokoi, and X. Zhang. Numerical methods for reachable space generation of humanoid robots. *The International Journal of Robotics Research*, 27(8):935–950, 2008. https://doi.org/10.1177/0278364908095142.

[51] T. Hamadneh. *Bounding Polynomials and Rational Functions in the Tensorial and Simplicial Bernstein Forms*. PhD thesis, Konstanz University, 01 2018.

[52] M. Hekmatnejad, G. Pedrielli, and G. Fainekos. Task scheduling with non-linear costs using smt solvers. In *2019 IEEE 15th International Conference on Automation Science and Engineering, CASE 2019*, pages 183–188, 8 2019. https://doi.org/10.1109/COASE.2019.8843048.

[53] Kunihiko Hiraishi. Solving optimiation problems on hybrid systems by graph exploration. *8th International Workshop on Discrete Event Systems*, 2006.

[54] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. https://doi.org/10.1016/0893-6080(89)90020-8.

[55] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Trans. Embed. Comput. Syst.*, 2019. https://doi.org/10.1145/3358228.

[56] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, pages 169–178, 2019.

[57] W. Jaimes, P. Acevedo, and V. Kafarov. Comparison of technology alternative for palm oil biodiesel production using exergy analysis. In Ian David Lockhart Bogle and Michael Fairweather, editors, *22nd European Symposium on Computer Aided Process Engineering*, volume 30 of *Computer Aided Chemical Engineering*, pages 207–211. Elsevier, 2012. https://doi.org/10.1016/B978-0-444-59519-5.50042-3.

[58] Y. F. Jin, Z. Y. Yin, W. H. Zhou, and H. W. Huang. Multi-objective optimization-based updating of predictions during excavation. *Engineering Applications of Artificial Intelligence*, 78:102–123, 2019.

[59] Joint FAO/WHO. *Standard for Quick Frozen Blocks of Fish Fillet, Minced Fish Flesh and Mixtures of Fillets and Minced Fish Flesh*, 1989. Codex Alimentarius.

[60] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, 2007. https://doi.org/10.1145/1275808.1276466.

[61] U. Kamath, J. Liu, and J. Whitaker. *Deep Learning for NLP and Speech Recognition*. Springer International Publishing, 2019. https://www.springer.com/gp/book/9783030145958.

[62] R. L. Karg and G. L. Thompson. A heuristic approach to solving travelling salesman problems. *Management science*, 10(2):225–248, 1964.

[63] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR*, abs/1702.01135, 2017. http://arxiv.org/abs/1702.01135.

[64] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. L. Dill, M. J. Kochenderfer, and C. Barrett. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification*, pages 443–452, Cham, 2019. Springer International Publishing. https://doi.org/10.1007/978-3-030-25540-4_26.

[65] O Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, 1986. https://doi.org/10.1177/027836498600500106.

[66] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. *Tik Report*, 216, 2006.

[67] P. Kouvaros and A. Lomuscio. Formal verification of cnn-based perception systems. *CoRR*, abs/1811.11373, 2018. http://arxiv.org/abs/1811.11373.

[68] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25:1097–1105, 01 2012. http://doi.org/10.1145/3065386.

[69] T. Langer. *On Generalized Barycentric Coordinates and Their Applications in Geometric Modeling*. PhD thesis, University of Saarlandes, 2008.

[70] K. G. Larsen and J. I. Rasmussen. Optimal reachability for multi-priced timed automata. *Theoretical Computer Science*, 390:197–213, 2008.

[71] N. Light and A. Walker. *Cook-chill catering: technology and management*. Springer Science & Business Media, 1990.

[72] Y. Lipman, J. Kopf, D. Cohen-Or, and D. Levin. Gpu-assisted positive mean value coordinates for mesh deformations. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, page 117–123, 2007.

[73] Changliu Liu, T. Arnon, C. Lazarus, C. W. Barrett, and M. J. Kochenderfer. Algorithms for verifying deep neural networks. *CoRR*, abs/1903.06758, 2019. http://arxiv.org/abs/1903.06758.

[74] M. Lombardi, M. Milano, and A. Bartolini. Empirical decision model learning. *Artificial Intelligence*, 244:343–367, 2017. https://doi.org/10.1016/j.artint.2016.01.005.

[75] A. Majumdar and D. Ghosh. Genetic algorithm parameter optimization using taguchi robust design for multi-response optimization of experimental and historical data. *International Journal of Computer Applications*, 127:27–32, 2015.

[76] MATLAB. *Global Optimization Toolbox User's Guide (R2019b)*. The MathWorks Inc., Natick, Massachusetts, 2019.

[77] E. Mehdizadeh, R. Tavakkoli-Moghaddam, and M. Yazdani. A vibration damping optimization algorithm for a parallel machines scheduling problem with sequence-independent family setup times. *Applied Mathematical Modelling*, 39(22):6845 – 6859, 2015.

[78] Minitab. *Statistical Software*. Minitab, Inc., State College, PA, 2018.

[79] R. G. Moreira. Impingement drying of foods using hot air superheated steam. *Journal of Food Engineering*, 49:291–295, 2011.

[80] S. Mudie, E. A. Essah, A. Grandison, and R. Felgate. Electricity use in the commercial kitchen. *International Journal of Low-Carbon Technologies*, 11(1):66–74, 2013. `https://doi.org/10.1093/ijlct/ctt068`.

[81] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

[82] A. Narkawicz, J. Garloff, A. Smith, and C. Muñoz. Bounding the range of a rational functiom over a box. *Reliable Computing*, 17, 12 2012.

[83] Truong Nghiem, Madhur Behl, Rahul Mangharam, and George Pappas. Green scheduling of control systems for peak demand reduction. In *IEEE Conference on Decision and Control and European Control Conference*, pages 5131–5136, 2011.

[84] A. Nguyen, S. Reiter, and P. Rigo. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058, 2014.

[85] M. Necati Özisik. *Heat Conduction*. Wiley, 1980.

[86] E. Paillat. Energy efficiency in food-service facilities: the case of långbro värdshus, 2011.

[87] S. Park and K. Miller. Random number generators: Good ones are hard to find. *Commun. ACM*, 31:1192–1201, 10 1988.

[88] M. Peleg and M. Corradini. Microbial growth curves: What the models tell us and what they cannot. *Critical reviews in food science and nutrition*, 51:917–45, 2011.

[89] V. Perdereau, C. Passi, and M. Drouin. Real-time control of redundant robotic manipulators for mobile obstacle avoidance. *Robotics and Autonomous Systems*, 41(1):41–59, 2002.

[90] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56:1247–1293, 2013.

[91] R. Roy, S. Hinduja, and R. Teti. Recent advances in engineering design optimisation: Challenges and future trends. *CIRP Annals - Manufacturing Technology*, 57:697–715, 2008.

[92] W. Ruan, X. Huang, and M. Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. *CoRR*, abs/1805.02242, 2018. `http://arxiv.org/abs/1805.02242`.

[93] G. Ruocco, M. V. De Bonis, and F. Marra. Combining microwave and jet impingement in a oven prototype. *Procedia Food Science*, 1:1331–1337, 2011.

[94] R. M. Rustamov. Boundary element formulation of harmonic coordinates. Technical report, Purdue University, 2007.

[95] T. Rybus. Obstacle avoidance in space robotics: Review of major challenges and proposed solutions. *Progress in Aerospace Sciences*, 101:31–48, 2018.

[96] M. Safeea, P. Neto, and R. Bearee. On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case. *Robotics and Autonomous Systems*, 119:278–288, 2019.

[97] A. Sarkar, N. Nitin, M. V. Karwe, and R. P. Singh. Fluid flow and heat transfer in air jet impingement in food processing. *Journal of Food Science*, 69(4):CHR113–CHR122, 2004.

[98] S. Shan and G.G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41:219–241, 2010.

[99] H. Shavandia, H. Mahlooji, and N. E. Nosratian. A constrained multi-product pricing and inventory control problem. *Applied Soft Computing*, 12:2454–2461, 2012.

[100] M. Stilman. Task constrained motion planning in robot joint space. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3074–3081, 2007.

[101] K. Tan, T. Lee, and E. Khor. Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. *Artificial Intelligence Review*, 17:251–290, 2002.

[102] M. Telgarsky. Neural networks and rational functions. *CoRR*, 2017.

[103] Emidio Tiberi. *Numerical modelling of thermal systems for professional food service appliances*. PhD thesis, Università degli Studi di Trieste, Academic Year 2017-2018. pp. 25-88.

[104] J. Titi and J. Garloff. Matrix methods for the simplicial bernstein representation and for the evaluation of multivariate polynomials. *Applied Mathematics and Computation*, 315:246–258, 12 2017. https://doi.org/10.1016/j.amc.2017.07.026.

[105] J. Titi and J. Garloff. Matrix methods for the tensorial bernstein form. *Applied Mathematics and Computation*, 346:254–271, 2019. https://doi.org/10.1016/j.amc.2018.08.049.

[106] J. Titi, T. Hamadneh, and J. Garloff. Convergence of the simplicial rational bernstein form. In *Modelling, Computation and Optimization in Information Systems and Management Sciences*, pages 433–441, Cham, 2015. Springer International Publishing.

[107] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson. NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. *CoRR*, abs/2004.05519, 2020.

[108] A. Tufano, R. Accorsi, F. Garbellini, and R. Manzini. Plant design and control in food service industry. a multi-disciplinary decision-support system. *Computers in Industry*, 103:72–85, 2018. https://doi.org/10.1016/j.compind.2018.09.007.

[109] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann. Manipulability analysis. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 568–573, 2012.

[110] D. A Van Veldhuizen and G. B. Lamont. On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, volume 1, pages 204–211. IEEE, 2000.

[111] M. Vlček. Chebyshev polynomial approximation for activation sigmoid function. *Neural Network World*, 22:387–393, 2012. https://doi.org/10.14311/NNW.2012.22.023.

[112] S. Waldron. Generalised affine barycentric coordinates. *Jaen Journal on Approximation*, 3, 2011.

[113] K. Wang, Q. Wang, Y. Cheng, H. Wu, Y. Song, and V. Bruno. Optimization of the geometric parameters of the east articulated maintenance arm (eama) with a collision-free workspace determination in east. *Fusion Engineering and Design*, 139:155–162, 2019. https://doi.org/10.1016/j.fusengdes.2019.01.033.

[114] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Formal security analysis of neural networks using symbolic intervals. *CoRR*, abs/1804.10829, 2018. http://arxiv.org/abs/1804.10829.

[115] J. Warren, S. Schaefer, A. Hirani, and M. Desbrun. Barycentric coordinates for convex sets. *Adv. Comput. Math.*, 27:319–338, 10 2007. https://doi.org/10.1007/s10444-005-9008-6.

[116] W. Xiang, H. Tran, and T. T. Johnson. Output reachable set estimation and verification for multilayer neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(11):5777–5783, 2018. https://arxiv.org/abs/1708.03322.

[117] K. Yang, M. Emmerich, A. Deutz, and C. M. Fonseca. Computing 3-d expected hypervolume improvement and related integrals in asymptotically optimal time. *Evolutionary Multi-Criterion Optimization*, pages 685–700, 2017.

[118] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 4939–4948. Curran Associates, Inc., 2018. https://proceedings.neurips.cc/paper/2018/file/d04863f100d59b3eb688a11f95b0ae60-Paper.pdf.

[119] An-Min Zou, Zeng-Guang Hou, Si-Yao Fu, and Min Tan. Neural networks for mobile robot navigation: A survey. In *Advances in Neural Networks - ISNN 2006*. Springer Berlin Heidelberg, 2006. http://doi.org/10.1007/11760023_177.

[120] W. Zou, Y. Zhu, H. Chen, and B. Zhang. Solving multiobjective optimization problems using artificial bee colony algorithm. *Discrerte Dynamics in Nature and Society*, 2011, 2011.