Evaluation of a Gradient Free and a Gradient Based Optimization Algorithm for Industrial Beverage Pasteurisation Described by Different Modeling Variants

Lea Miko Versbach

Master's thesis
2016:E11

Lund University

Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

Centrum Scientiarum Mathematicarum

Master's Theses in Mathematical Sciences 2016:E11
ISSN 1404-6342

LUNFNA-3021-2016

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/

Master's Thesis

Evaluation of a Gradient Free and a Gradient Based Optimization Algorithm for Industrial Beverage Pasteurisation Described by Different Modeling Variants

Utvärdering av en Derivata-Fri och en Gradient-Baserad Optimeringsalgoritm för Industriell Pastörisering av Drycker Beskrivna med Olika Modelleringsvarianter

Lea Miko Versbach

June 10, 2016

Academic supervisor: Professor Claus Führer

Industrial supervisor: Dr. Falko Jens Wagner, Krones Group

Examinator: Dr. Andrey Ghulchak

Opponents: Matilda Hjälle, Robert van Heyningen

Abstract

Inspired by Krones Group in Holte/Copenhagen, the intention of this thesis is to simulate and optimize pasteurisation processes. Based on different modes of heat transfer, a mathematical model to describe the thermal processes in the product is developed. The main goal of the thesis is to optimize the thermal treatment of the product. Both the derivative-free COBYLA and the gradient-based MMA optimization algorithm are described and evaluated. The optimal results obtained with the NLopt library in Python are compared and different modeling variants of the optimization problem are developed with regard to their meaningfulness and their numerical behaviour.

Acknowledgments

I would like to express my gratitude towards my supervisors, Professor Claus Führer and Dr. Falko Jens Wagner.
Thank you Claus, for helping me more than I ever expected. Thanks for showing me how much fun it is writing my Master's Thesis and being open-minded for any kind of side-projects and playgrounds we detected during the last months. It is impossible to describe how much I learned from you - vielen Dank!
Thank you Falko, for giving me the chance to make first steps from University towards industry and giving me an insight into the work of an mathematician outside of university. I hope to get the chance working with you again in the future.
Thanks to Stefan Kaatz for inspiring me with your great Bachelor thesis.
I would also like to express my gratitude to Lund University for providing me with a place to work. Thanks to the fellow Master students sitting with me in the Da Vinci room and the others, who shared time at fika and lunch with me - without your support I would not have enjoyed the last months that much! It was great being able to discuss problems concerning my work and this report, but also having a supporting and happy working atmosphere. A special thanks to Marcus for helping me to give this thesis a Swedish title.
And finally, thanks to my family for your emotional support and believing in me and my mathematical skills even when I myself had doubts.

Contents

1 Introduction

It is very common nowadays to buy beverages or canned food which do not need to be cooled when they are stored. Companies can provide their product with an expected expiration date since it is processed during the food production procedure. This is achieved by thermal treatment in the production chain, which allows to extend the product's shelf life by inactivating specific micro-organisms in a natural way without the use of chemical extras.

The Krones Group, a company with headquater in Neutraubling, Germany, plans, develops and manufactures machines and complete lines for the fields of process, filling and packaging technology. Every fourth beverage worldwide is produced, filled or packaged on lines from Krones.
Krones' product portfolio also covers the pasteurisation technology which is the thermal treatment of products. In order to provide a gentle heat treatment, this process field needs to be optimized.

In Chapter 2, basics about pasteurisation are introduced. The historic idea of thermal treatment is discussed and modern pasteurisation techniques and machines are presented.
Chapter 3 serves to give an overview over the physical processes in heat transfer. For the example of a can, the modes of heat transfer in the pasteurisation process are visualized. Based on these modes of heat transfer, both a static and a dynamic mathematical model to describe the pasteurisation process are developed in Chapter 4. The model is simplified by some assumptions in order to reduce the computational effort.
In Chapter 5, both the gradient-free COBYLA and the derivative-based MMA optimization algorithm are presented and their implementation in the NLopt library in Python is discussed.
These optimization routines are applied in Chapter 6 to optimize the pasteurisation process using the mathematical model developed in Chapter 4. Based on the results of optimizing the thermal treatment, different modeling variants of the optimization problem are developed to improve the exactness of the optimal solution with respect to the computational effort of the algorithms.
In order to prepare an extension of the modeling detail level, in Chapter 7 the governing thermodynamic formulation of heating a liquid in a container is set up. Some numerical approaches are studied. The coupling of this model with optimization methods is a topic for future work.
In Chapter 8, the conclusion of the work in this thesis is presented. Moreover, ideas for future work detected in this thesis are introduced.

2 Fundamentals

The intent of this chapter is to give a short overview of food production processes and a method to extend the product's minimal shelf life by thermal treatment. This background knowledge is the basis to understand the application of the mathematical results for the Krones Group, which is the core of this thesis.

2.1 Pasteurisation

Any product contains micro-organisms, widely known as bacteria. Some of them are useful or even needed, for example in milk products. Others might be dangerous when consumed or make the product go bad quickly. This motivates the need of a method to inactivate specific micro-organisms in the food production industry.

In 1864, the French chemist Louis Pasteur discovered the positive effect of heat treating products to reduce the number of micro-organisms [10, p.501]. Pasteur was one of the first to notice the thermal inactivation of micro-organisms, thus this method is known as pasteurisation nowadays.

Definition 2.1.1 (D-value)**.** Heating up a homogeneous population to a constant lethal temperature will result in a decreasing number of survivors. Mathematically, this process is decribed in [17, p.5] by

$$N_t = N_0 10^{-\frac{t}{D}}. \tag{2.1}$$

N_t denotes the number of surviving organisms at time t, N_0 the initial number of viable organisms and t the time of heat treatment in minutes. The D-value D characterizes the decimal reduction time. This is the required time in minutes to reduce the specific population to ten percent.

The D-value is dependent on the specific micro-organism considered. Tests indicated a growing lethal rate of the micro-organisms with increasing temperature; in consequence, the D-value decreases. This motivates the introduction of a second constant which depends on the temperature.

Definition 2.1.2 (z-value)**.** The z-value describes the dependence of the D-value on the temperature. It states how much the temperature in °C has to be changed to get a ten times higher lethal effect. This is equivalent to reducing the D-value to one tenth.

The purpose of pasteurisation is not to inactivate all micro-organisms and consequently to sterilize the product. The pasteurisation process aims to reduce the number of micro-organisms in a product to a level of biological stability. At the same time, the heat

treatment must neither change the quality of the final product, nor affect its specification like taste or consistency. The number of micro-organisms has to be reduced with a minimal change in the aroma. Therefore, an optimized and gentle heat treatment is of interest, [17, p.3].
Louis Pasteur's experiments indicated 60 °C being an suitable lethal temperature, which explains the historic definition to measure pasteurisation effects.

Definition 2.1.3 (PU, historical)**.** One Pasteurisation Unit (PU) corresponds to one minute of heating at 60 °C.

Nowadays, a product specific definition is preferred. The number of PUs reached in a heat treated product depends on the product and the particular micro-organisms it contains.

Definition 2.1.4 (PU, contemporary)**.** The change of PUs with time and the product temperature u_p measured in °C are connected by an exponential relationship:

$$\frac{\mathrm{dPU}}{\mathrm{d}t} = 10^{\frac{u_p(t)-u_{\mathrm{ref}}}{z}}. \tag{2.2}$$

u_{ref} is a given reference temperature for the product-specific z-value z, both given in °C.

The exact number of PUs can only be obtained by measurements. Nevertheless, the process can be controlled theoretically by integrating Equation (2.2) over the given time interval [17, p.11].

Definition 2.1.5 ($\mathrm{PU}_{\mathrm{end}}$)**.** The amount of PUs in a product after the thermal treatment in the time interval $[t_0, t_{\mathrm{end}}]$ is given by

$$\mathrm{PU}_{\mathrm{end}} = \int_{t_0}^{t_{\mathrm{end}}} \mathrm{PU}_{\mathrm{co}}(t)dt \tag{2.3}$$

with

$$\mathrm{PU}_{\mathrm{co}}(t) = \begin{cases} 10^{\frac{u_p(t)-u_{\mathrm{ref}}}{z}} & , u_p \geq u_{\mathrm{CutOff}} \\ 0 & , u_p < u_{\mathrm{CutOff}}. \end{cases} \tag{2.4}$$

The temperature of the product at time t is given by u_p, u_{ref} describes the temperature reference for the product specific z-value z. u_{CutOff} is the product specific cut-off temperature and illustrates the minimal temperature to achieve lethal effects on the micro-organism, all measured in °C.

2.2 Canning Operations

The process of thermal food treatment is one of several operations in the production process. Chart 2.1 gives an overview of the different operations taking place. The process can be divided into several typical steps [13, p.2 ff.]:

1: Selecting suitable incredients and preparing the product.

2: Filling the product into containers and sealing the containers.

3: Pasteurising the product.

4: Storing the canned product at a suitable temperature.

5: Labelling, packing, distribution, marketing and consumption of the canned product.

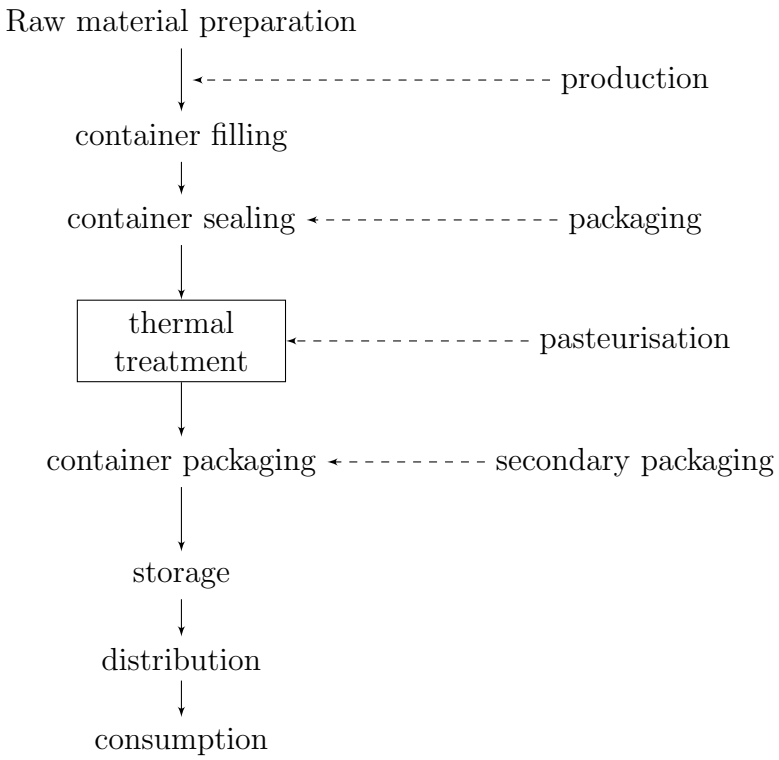


Figure 2.1: Food production line

2.3 Pasteurs

There exist several techniques for the thermal treatment of the product. The main alternatives are to pasteurise the product either before or after it is filled into the container. In a plate pasteur, the product is pasteurised before it is filled into the container, while it is pasteurised after it is filled into the container in a tunnel pasteur. Both methods are presented in the next sections. For the rest of the thesis it is assumed that the product is pasteurised in a tunnel pasteur. Therefore, its construction and mode of action is discussed in more detail.

2.3.1 Plate Pasteur

In a plate pasteur, the product is heat treated before it is filled into the container. This pasteurisation technique was invented by Louis Pasteur and patented by him in 1873. The disadvantage of the method is the risk of a possible reinfection of the product. Nevertheless, plate pasteurs are used for example in the case of containers not able to handle temperatures above 60 °C or beverages stored in kegs. As this thesis only considers tunnel pasteurs, the reader is referred to [17, p.60 ff.] for further details.

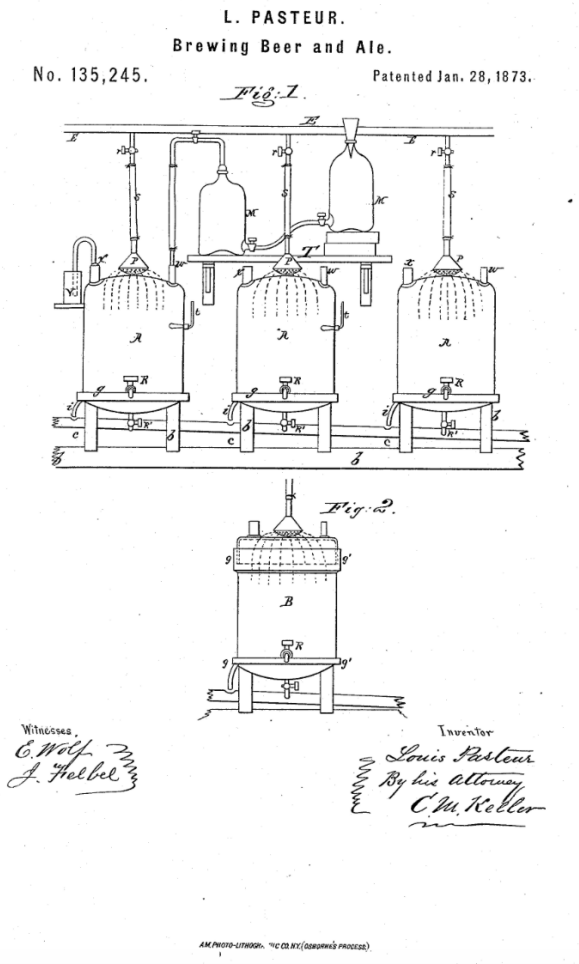


Figure 2.2: Patent for a plate pasteur by Louis Pasteur in 1873, [5]

2.3.2 Tunnel Pasteur

A tunnel pasteur is one sector of several processing zones. These zones are connected by a conveyor belt which carries the product automatically through each zone. Once the container is filled with the product and sealed, it is transported through a tunnel on this conveyor belt. Here, the product is heated, pasteurised and afterwards cooled. This

is achieved by spraying water of different temperatures over the container. Zones with different spray water temperatures are separated by gaps of 20 cm to prevent a mixture of the spray water and to avoid sudden temperature changes. Zones of temperature high enough to achieve pasteurisation processes are called pasteurisation zones, the heating and cooling zones are referred to as recuperation zones. The pasteurisation zones are not separated by gaps since the temperature difference between these zones is small enough to ignore the water mixture. This construction of a tunnel pasteur provides a progressively heating and cooling of the product. For energy reasons, each heating zone is connected with a cooling zone to reuse spray water of a similar temperature. This construction is illustrated in Figure 2.3.

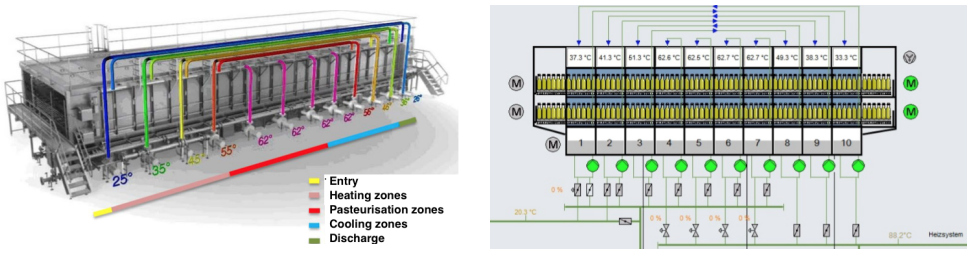


Figure 2.3: Construction of a standard tunnel pasteur, [1]

The transportation of the product through the tunnel pasteur can be done in two different ways: Walking beam systems have the advantage of being able to handle fragments of broken containers, useful in the case of glass bottles. The flat bed system uses a metal or plastic belt to fix the container.
Tunnel pasteurs can be single, double or tripple deck machines. The advantage of more-deck machines is the increased use of floor area capacity and the use of the same water system.

The LinaFlex tunnel pasteur constucted by Krones can be seen in Figure 2.4. It can treat up to 70,000 containers per hour in the single-deck version and up to 180,000 containers per hour in the double-deck version. The length of the machine can be up to 40 m and there exist different treatment widths between approximately 3.27 m and 5.73 m. [1]

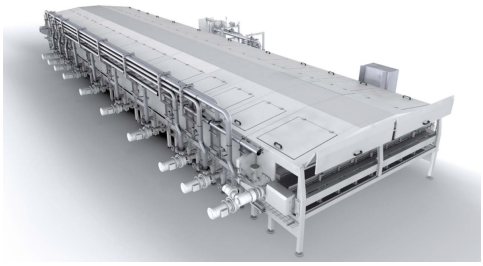


Figure 2.4: LinaFlex tunnel pasteur constructed by Krones, [1]

3 Heat Transfer

For the rest of the thesis, a container describes either a glass/PET-bottle or a can filled with a liquid product. For simplicity it is assumed that the container holds beer unless stated otherwise.

It is important to analyze the effect of heat treatment by spray water on the canned product in order to improve the understanding of the pasteurisation process. The term heat is defined in [7, p.218] by means of temperature.

Definition 3.0.1 (Temperature)**.** Temperature describes how hot an object is and is measured in degree Celsius (°C) or Kelvin (K), with [°C] + 273.15 = [K].

The temperature of an object changes due to the heat energy transferred between different states of temperature.

Definition 3.0.2 (Heat)**.** Heat is a form of energy and measured in Joules (J).

Heat is always transferred from regions of higher temperature to regions of lower temperature.

3.1 Modes of Heat Transfer

There exist three fundamental mechanisms of heat transfer [17, p.17]:

- **Conduction** is the transfer of heat through a stationary body, either solid, liquid or gas. No movement occurs within the body except of that between atoms and molecules, called molecular motion [13, p.14].
- **Convection** is the transfer of heat in fluid systems as a result of movement within the system, called fluid flow [13, p.14]. The movement is either forced or natural and causes a mixing within the system.
- **Radiation** is the transfer of heat by means of electromagnetic waves between bodies of different temperatures without having physical contact [13, p.14].

More than one mode of heat transfer can occur throughout thermal processes. During the pasteurisation in a tunnel pasteur, several modes of heat transfer can be observed. In the next section, the heat transfer processes in a container filled with a liquid product are described in more detail.

3.2 Physical Process of Heating Liquids in a Container

Both conductive and convective heat transfer are involved when liquids in a container are heated or cooled [13, p.14 ff.]:

- Heat transfer from the surrounding to the container is described by convection.
- Heat transfer through the container wall to the product is described by conduction.
- Heat transfer within the liquid product is described by convection.

Figure 3.1 shows these different modes of heat transfer for the example of a cylindric can filled with a liquid product.

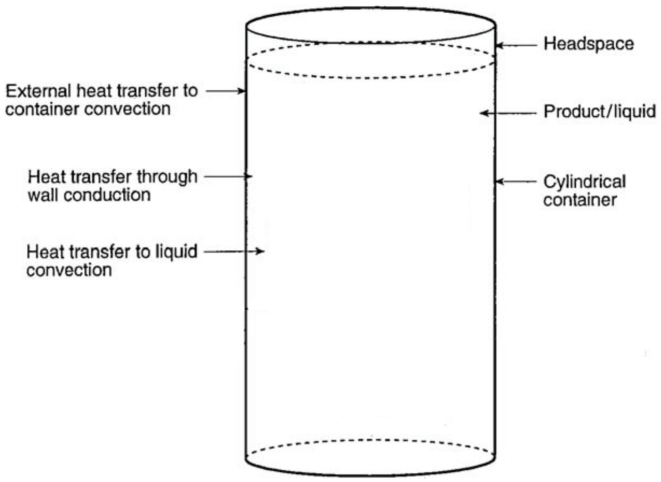


Figure 3.1: Simplified model of the different modes of heat transfer in a container holding a liquid product, [13, p.15]

Assuming both the container and the liquid have a constant temperature and are heated from the outside by spray water, heat energy is transferred through the heated container to the regions of liquids closest to the container wall. As a result of the energy change, the density of the molecules decreases and the heated fluid region rises until it reaches the top of the liquid. The induced fluid motion causes it to fall in the central core and the hot fluid next to the wall is replaced by colder fluid in the core. This convective heating process is sketched in Figure 3.2. During the cooling process, the fluid motion is reversed. The density of the molecules next to the container wall increases and the cooled fluid region falls until it reaches the bottom of the container. The temperature change at the bottom of the container causes it to rise in the central core, and colder fluid next to the wall is replaced by warmer fluid in the core.
With proceeding time, the temperature of the product becomes more uniform, the energy difference smaller and the fluid velocity tends to decrease. Once a thermal equilibrium is reached, the fluid motion will discontinue, [13, p.45 ff.].

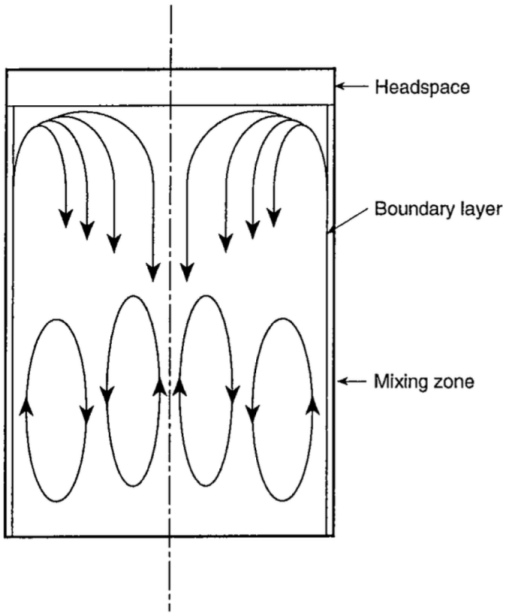


Figure 3.2: Simple model of the physical heat process in a container, [13, p.47]

4 Mathematical Model

When heat energy is applied to an object which is assumed to have a homogeneous temperature distribution, its temperature will change due to the physical processes described in the previous chapter. There exists different approaches to model the heating process of the product mathematically. [7, p.218 ff.]

Since greater mass absorbs more energy than lesser mass, a first approach takes into account the mass of the object. The mathematical model is obtained by assuming the heat flow with time being directly proportional to the temperature change and the mass of the object:

$$\frac{\mathrm{d}Q}{\mathrm{d}t} = cm\frac{\mathrm{d}u}{\mathrm{d}t}, \tag{4.1}$$

with Q the heat flow with time in W, m the mass of the object in kg and u the temperature of the object in °C. The specific heat energy of the material, i.e. the needed amount of heat energy to raise the temperature of 1 kg of the given substance at a given temperature by 1 °C, is described by c and measured in kJ/(kg °C). The specific heat energy is not constant over a large temperature range.

A second approach takes into account that the heat energy of an object increases or decreases if it exchanges heat energy with its surrounding. The heat flow between the object and the surrounding depends on the surface of the object. In consequence, the heat flow depends on the surface area of the object being able to interact with the surrounding. Newton's law of cooling can be applied assuming the heat flow being directly proportional to the temperature difference between the surface and its immediate surrounding:

$$\frac{\mathrm{d}Q}{\mathrm{d}t} = hA(u(t) - u_s(t)), \tag{4.2}$$

with h the conductive heat transfer coefficient in W/(m² °C) and A the surface area from which heat energy is lost or gained in m². u_s is the temperature of the surrounding and u the temperature of the object's surface and interior, both measured in °C.

Combining Equations (4.1) and (4.2) gives a differential equation to describe the temperature variation of the object with time, depending on the object's mass and surface area:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{hA}{cm}(u(t) - u_s(t)) \tag{4.3}$$

4.1 Static Model

Equation (4.3) is used by Krones to mathematically describe the processes of heating a container with liquids in a tunnel pasteur. Some fundamental simplifications are made; it is assumed that the container holds a perfectly mixed product. Due to this assumption, the convective heat transfer processes occuring in the liquid can be ignored. Thus, any point in the product can be used for measuring the product's temperature, and the shape of the container can be neglected.
Let u_b be the body temperature and u_s the spray water temperature. Equation (4.3) becomes

$$\frac{\mathrm{d}u_b(t)}{\mathrm{d}t} = \frac{hA}{mc_p}(u_s(t) - u_b(t)) = \kappa(u_s(t) - u_b(t)) \tag{4.4}$$

for $\kappa = \frac{hA}{mc_p}$.
The differential equation (4.4) can be solved by Laplace transformation. Some basics about Laplace transformation are dicussed now, more details can be found in [6, p.207 ff.].

4.1.1 Basics About Laplace Transformation

Definition 4.1.1. The Laplace transformation $\mathcal{L}$ of a real valued function f on $[0, \infty)$ is defined by

$$\mathcal{L}\{f(t)\}(s) = \int_0^\infty e^{-st} f(t)dt \tag{4.5}$$

provided $\lim_{r\to\infty} \int_0^r e^{-st} f(t)dt$ exists and is finite.

An important property of the Laplace transformation is its linearity.

Proposition 4.1.1. The Laplace transformation $\mathcal{L}$ is linear, that is for f and g real valued functions defined on $[0, \infty)$ and $a, b \in \mathbb{R}$,

$$\mathcal{L}\{af(t) + bg(t)\}(s) = a\mathcal{L}\{f(t)\}(s) + b\mathcal{L}\{g(t)\}(s) \tag{4.6}$$

for each s such that the right hand side makes sense.

Proof.

$$\begin{aligned}\mathcal{L}\{af(t) + bg(t)\}(s) &= \int_0^\infty e^{-st}(af(t) + bg(t))dt \\ &= a\int_0^\infty e^{-st} f(t)dt + b\int_0^\infty e^{-st} g(t)dt \\ &= a\mathcal{L}\{f(t)\}(s) + b\mathcal{L}\{g(t)\}(s).\end{aligned}$$

□

The next theorem can be applied to solve differential equations by means of Laplace transformation.

Theorem 4.1.1. Suppose f is differentiable for $t \geq 0$ and is of exponential order, that is $|f(t)| \leq Me^{\alpha t}$ for some constants M and α for all $t \geq 0$. If $\mathcal{L}\{f'(t)\}(s)$ exists, then

$$\mathcal{L}\{f'(t)\}(s) = s\mathcal{L}\{f(t)\}(s) - f(0). \tag{4.7}$$

Proof. By definition $\mathcal{L}\{f'(t)\}(s) = \int_0^\infty e^{-st} f'(t)dt$. Integration by parts gives

$$\int_0^r e^{-st} f'(t)dt = e^{-sr} f(r) - f(0) + s \int_0^r e^{-st} f(t)dt.$$

Since f is of exponential order, $e^{-sr} f(r) \to 0$ for $r \to \infty$. This implies $\mathcal{L}\{f'(t)\}(s) = s\mathcal{L}\{f(t)\}(s) - f(0)$. □

Another important property of the Laplace transformation is the existence of its inverse.

Definition 4.1.2. Let F be given. If there exists a function f such that $\mathcal{L}\{f(t)\}(s)$ exists and $\mathcal{L}\{f(t)\}(s) = F(s)$, then F has an inverse Laplace transformation given by f. In this case, $f(t) = \mathcal{L}^{-1}\{F(s)\}(t)$.

The inverse Laplace transformation is also linear. This can be deduced from the linearity result for the Laplace transformation.

Definition 4.1.3. The convolution of two piecewise continuous functions f and g on $[0, \infty)$ is defined by

$$(f * g)(t) = \int_0^t f(\tau)g(t - \tau)d\tau.$$

Convolution and Laplace transformation have an important relation.

Proposition 4.1.2. Let f, g be piecewise continuous and of exponential order. For $F(s) = \mathcal{L}\{f\}$ and $G(s) = \mathcal{L}\{g\}$ it follows

$$\mathcal{L}\{(f * g)(t)\}(s) = F(s)G(s), \tag{4.8}$$
$$\mathcal{L}^{-1}\{F(s)G(s)\}(t) = (f * g)(t). \tag{4.9}$$

Proof.

$$\begin{aligned}
F(s)G(s) &= \Big(\int_0^\infty e^{-sx} f(x)dx\Big)\Big(\int_0^\infty e^{-s\tau} g(\tau)d\tau\Big) \\
&= \int_0^\infty \Big(\int_0^\infty e^{-s(x+\tau)} f(x)dx\Big) g(\tau)d\tau \\
&= \int_0^\infty \Big(\int_\tau^\infty e^{-st} f(t - \tau)dt\Big) g(\tau)d\tau \\
&= \int_0^\infty \Big(\int_\tau^\infty e^{-st} f(t - \tau) g(\tau)dt\Big) d\tau \\
&= \int_0^\infty e^{-st} \Big(\int_0^t f(t - \tau) g(\tau)d\tau\Big) dt \\
&= \mathcal{L}\{(f * g)(t)\}(s).
\end{aligned}$$

Applying the inverse Laplace transformation, the Proposition is proved. □

4.1.2 Solution of the Static Model

The properties of the Lapalace transformation allow to express the solution of Equation (4.4) as

$$u_b(t) = u_0 e^{-\kappa t} + \kappa \int_0^t e^{-\kappa(t-\tau)} u_s(\tau) d\tau. \tag{4.10}$$

The present model only takes into account the heat flow between the spray water and the product, ignoring the effect of the container material on the energy flow. A simple modification allows to consider separately the heat flow between the spray water and the container, and the heat flow between the container and the product. These heat flows depend on each other, and are given by the differential equations

$$\frac{\mathrm{d}u_c(t)}{\mathrm{d}t} = \kappa_1(u_s(t) - u_c(t)), \tag{4.11}$$

and

$$\frac{\mathrm{d}u_p(t)}{\mathrm{d}t} = \kappa_2(u_c(t) - u_p(t)). \tag{4.12}$$

u_s describes the spray water temperature, u_c the container temperature and u_p the product temperature in °C. κ_2 and κ_2 are product specific constants in s^{-1}.
The solutions of (4.11) and (4.12) are given by

$$u_c(t) = u_c(0) e^{-\kappa_1 t} + \kappa_1 \int_0^t e^{-\kappa_1(t-\tau)} u_s(\tau) d\tau \tag{4.13}$$

and

$$u_p(t) = u_p(0) e^{-\kappa_2 t} + \kappa_2 \int_0^t e^{-\kappa_2(t-\tau)} u_c(\tau) d\tau \tag{4.14}$$

for initial conditions $u_c(0)$ and $u_p(0)$.
In a tunnel pasteur, the velocity of the conveyor belt is very slow. As a result, the spray water temperature u_s and in consequence the container temperature u_c are constant for small time intervals $[0, t]$. This assumption allows to solve Equations (4.13) and (4.14):

$$u_c(t) = u_c(0) e^{-\kappa_1 t} + u_s(0)(1 - e^{-\kappa_1 t}) \tag{4.15}$$

$$u_p(t) = u_p(0) e^{-\kappa_2 t} + u_c(0)(1 - e^{-\kappa_2 t}) \tag{4.16}$$

In the model implementation, the temperatures of the container and the product is calculated incrementally for small time intervals Δt at times t_n. Equations (4.15) and (4.16) are reformulated by taking into account the changing initial condition of the differential equations, depending on the last temperatures calculated. This gives the equations for the final product model:

$$u_c(t_n) = u_c(t_{n-1}) e^{-\kappa_1 \Delta t} + u_s(t_{n-1})(1 - e^{-\kappa_1 \Delta t}) \tag{4.17}$$

$$u_p(t_n) = u_p(t_{n-1}) e^{-\kappa_2 \Delta t} + u_c(t_{n-1})(1 - e^{-\kappa_2 \Delta t}) \tag{4.18}$$

This model is a variation between steady and non-steady temperature equations. The time constants for the heating and cooling processes are considered independently by

$$\kappa_1 = \begin{cases} H_1, & \text{if } u_c \leq u_s \\ C_1, & \text{else} \end{cases} \tag{4.19}$$

and

$$\kappa_2 = \begin{cases} H_2, & \text{if } u_p \leq u_c \\ C_2, & \text{else.} \end{cases} \tag{4.20}$$

The values κ_1 and κ_2 depend on the product and are obtained by measurements. This is done by heat treating the product in the tunnel pasteur with a thermometer inside the container.

4.1.3 Example for Standard Pasteur

The model derived in the last section is applied to a standardized pasteur and a standardized product. The corresponding values can be found in the following tables and are used to plot Figure 4.1. The influence of the container temperature on the product temperature can be seen in Figure 4.1 since both functions have a very similar curvature. Moreover, the product temperature is slightly lower than the container temperature due to the mathematical model and the heat loss during the conductive heat transfer through the container material. After the product reaches a temperature of 58 °C, first pasteurisation effects can be noticed. The amount of PUs increases as long as the product temperature is above the cut-off temperature of 58 °C, then it ceases.

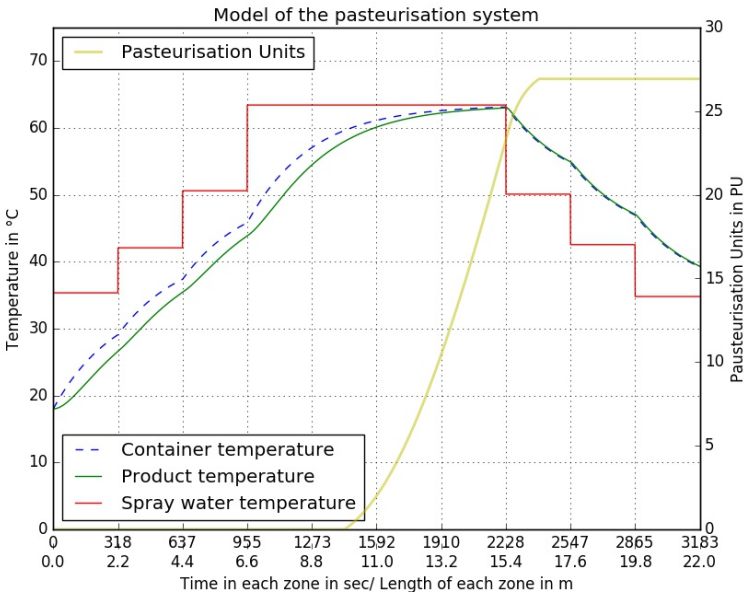


Figure 4.1: Krones AG Model for standardized product and pasteur

Number of heating zones	3
Number of pasteurisation zones	4
Number of cooling zones	3
Number of subzones $z_{S,N}$	440
Lenght of a subzone l_U	0.05 m
Length of each zone	2.2 m
Lenght of the machine l_M	22.0 m
Width of each zone l_w	5.13 m
High of the tunnel l_h	1.221 m
Heat flow between zones without gap	1.36 $\mathrm{kWm^{-2}K^{-1}}$
Heat flow between zones with gap	0.54 $\mathrm{kWm^{-2}K^{-1}}$

Table 4.1: Data for the standardized tunnel pasteur

Product	beer
Type of container	bottle
Diameter	75 mm
Weight of the container	474 g
Volume	630 ml
Products pasteurised per hour	25000
Desired PU	25 PU
Z-value	6.94 °C
Cut-off temperature u_{CutOff}	58 °C
Initial temperature $u_p(0) = u_c(0)$	18 °C
Velocity of the voyager v_v	0.006911 $\mathrm{ms^{-1}}$
H_1	0.0032 $\mathrm{s^{-1}}$
H_2	0.0105 $\mathrm{s^{-1}}$
C_1	0.0032 $\mathrm{s^{-1}}$
C_2	0.136 $\mathrm{s^{-1}}$

Table 4.2: Data for the standardized product

4.2 Dynamic Model

The model considered so far is a static model. Passed time since the product entered the tunnel pasteur and position of the product in the tunnel depend on each other and are connected via the velocity of the conveyor belt. A dynamic model becomes important as the conveyor belt can have sudden stops. In consequence, the product will remain in the tunnel pasteur for a longer time. Stops of the conveyor belt can occur due to different reasons, mostly other processing zones have problems and the buffer sector between the zones are completed.
The static model for the heat conduction is given by Equation (4.4). In this case, time and position depend on each other. The dependence is given by the constant conveyor

belt velocity v, therefore the temperature functions only depend on time t.
In the dynamic case, time t and position z are independent of each other. By applying the linear transport equation, the dynamic heat equation is given by

$$\frac{\partial u_p(z,t)}{\partial t} + v\frac{\partial u_p(z,t)}{\partial z} = -\kappa(u_s(z,t) - u_p(z,t)). \tag{4.21}$$

The initial condition describes the temperature of the product at time $t = 0$ depending on its position in the tunnel pasteur and is denoted by $u_p(z,0) = u_0(z)$. The boundary condition describes the temperature of the product once it enters the tunnel pasteur and is given by $u_p(0,t) = u_{\text{in}}$.
The dynamic equation can be solved by a change of variables to transfer the PDE into an ODE and solving the ODE instead. This is called Method of Characteristics and its basic idea is illustrated in the next section.
The solution depends on the initial and boundary conditions. Which one is to be applied depends on the passed time since the product entered the tunnel pasteur.

- $t \leq \frac{z}{v} \Leftrightarrow z \geq vt$: The initial condition $u_p(z,0) = u_0(z)$ holds.
- $t > \frac{z}{v} \Leftrightarrow z < vt$: The boudary condition $u_p(0,t) = u_{\text{in}}$ holds.

4.2.1 Method of Characteristics

The method of characteristics is a technique to solve partial differential equations by a coordinate transformation. Curves are discovered on which the partial differential equation becomes an ordinary differential equation. This ordinary differential equation can be solved by transforming the coordinates back and a solution for the partial differential equation is found. The curves on which the partial differential equation is transformed into an ordinary differential equation are called characteristics. [12, p.97 ff.]
The coordinates x and t of the partial differential equation

$$P(x,t,u)\frac{\partial u}{\partial t} + Q(x,t,u)\frac{\partial u}{\partial x} = R(x,t,u),$$

are parametrized by the new coordinates τ in t-direction and ξ in x-direction, i.e. $t = t(\tau,\xi)$ and $x = x(\tau,\xi)$. Then $u(x,t) = u(x(\tau,\xi),t(\tau,\xi))$ and differentiating gives

$$\frac{\partial u}{\partial \tau} = \frac{\partial u}{\partial t}\frac{\partial t}{\partial \tau} + \frac{\partial u}{\partial x}\frac{\partial x}{\partial \tau}.$$

Comparing this with the original partial differential equation leads to

$$\begin{aligned}
\frac{\partial t}{\partial \tau} &= P(x,t,u),\\
\frac{\partial x}{\partial \tau} &= Q(x,t,u),\\
\frac{\partial u}{\partial \tau} &= R(x,t,u).
\end{aligned}$$

This system of ordinary differential equations can be solved easier than the original partial differential equation.

4.2.2 Solution of the Simple Dynamic Heat Model

Position and time are given by $z = z(\eta, \sigma)$ and $t = t(\eta, \sigma)$ when the coordinates η and σ are introduced. Equation (4.21) becomes

$$\begin{aligned}\frac{\partial u_p(z(\eta,\sigma),t(\eta,\sigma)))}{\partial t} + v\frac{\partial u_p(z(\eta,\sigma),t(\eta,\sigma)))}{\partial z} = \\ -\kappa(u_s(z(\eta,\sigma),t(\eta,\sigma))) - u_p(z(\eta,\sigma),t(\eta,\sigma)))).\end{aligned} \tag{4.22}$$

In the case $t \leq \frac{z}{v}$, comparison with equation $\frac{\partial u_p}{\partial \sigma} = \frac{\partial u_p}{\partial z}\frac{\partial z}{\partial \sigma} + \frac{\partial u_p}{\partial t}\frac{\partial t}{\partial \sigma}$ gives

$$\frac{\partial t}{\partial \sigma} = 1,\ \frac{\partial z}{\partial \sigma} = v,\ \frac{\partial u_p}{\partial \sigma} = -\kappa(u_s(z(\eta,\sigma),t(\eta,\sigma) - u_p(z(\eta,\sigma),t(\eta,\sigma)))).$$

With the initial conditions $t(\eta, 0) = 0$, $z(\eta, 0) = \eta$ and $u_P(\eta, 0) = u_0(\eta)$, the solutions of the ordinary differential equations are given by

$$\begin{aligned}t(\eta,\sigma) &= \sigma,\\ z(\eta,\sigma) &= \eta + v\sigma,\\ u_p(\eta,\sigma) &= u_0\eta e^{-\kappa\sigma} + \kappa\int_0^\sigma u_s(\eta + vs, s)e^{-\kappa(\sigma-s)}ds.\end{aligned}$$

As $\eta = z - vt$ and $\sigma = t$, the function for the product temperature in the dynamic case is given by

$$u_p(z,t) = u_0(z - vt)e^{-\kappa t} + \kappa\int_0^t u_s(z - v(t-s), s)e^{-\kappa(t-s)}ds. \tag{4.23}$$

In the case $t > \frac{z}{v}$, comparison with equation $\frac{\partial u_p}{\partial \eta} = \frac{\partial u_p}{\partial z}\frac{\partial z}{\partial \eta} + \frac{\partial u_p}{\partial t}\frac{\partial t}{\partial \eta}$ gives

$$\frac{\partial t}{\partial \eta} = 1,\ \frac{\partial z}{\partial \eta} = v,\ \frac{\partial u_p}{\partial \eta} = -\kappa(u_s(z(\eta,\sigma),t(\eta,\sigma) - u_p(z(\eta,\sigma),t(\eta,\sigma)))).$$

With the boundary conditions $t(0, \sigma) = \sigma$, $z(0, \sigma) = 0$ and $u_p(0, \sigma) = u_{\text{in}}$, the solutions of the ordinary differential equations are given by

$$\begin{aligned}t(\eta,\sigma) &= \sigma + \eta,\\ z(\eta,\sigma) &= v\eta,\\ u_p(\eta,\sigma) &= u_{\text{in}}e^{-\kappa\eta} + \kappa\int_0^\eta u_s(vs, s)e^{-\kappa(\eta-s)}ds.\end{aligned}$$

As $\eta = \frac{z}{v}$ and $\sigma = t - \frac{z}{v}$, the function for the product temperature in the dynamic case is given by

$$u_p(z,t) = u_{\text{in}}e^{-\kappa\frac{z}{v}} + \kappa\int_0^{\frac{z}{v}} u_s(vs, s)e^{-\kappa(\frac{z}{v}-s)}ds. \tag{4.24}$$

Therefore, the dynamic product temperature function is given by

$$u_p(z,t) = \begin{cases} u_0(z - vt)e^{-\kappa t} + \kappa\int_0^t u_s(z - v(t-s), s)e^{-\kappa(t-s)}ds, & \text{if } t \leq \frac{z}{v} \\ u_{\text{in}}e^{-\kappa\frac{z}{v}} + \kappa\int_0^{\frac{z}{v}} u_s(vs, s)e^{-\kappa(\frac{z}{v}-s)}ds, & \text{if } t > \frac{z}{v}. \end{cases} \tag{4.25}$$

4.2.3 Improved Dynamic Heat Model

The improved heat model distinguishes between the spray water temperature u_s, the container temperature u_c and the product temperature u_p. The values for κ_1 and κ_2 are given by

$$\kappa_1 = \begin{cases} H_1, & u_c \le u_s \\ C_1, & \text{else} \end{cases} \tag{4.26}$$

and

$$\kappa_2 = \begin{cases} H_2, & u_p \le u_c \\ C_2, & \text{else.} \end{cases} \tag{4.27}$$

The change of PUs in time is described by

$$\frac{\mathrm{dPU}(u_p(z,t))}{\mathrm{d}t} = \mathrm{PU}_{\mathrm{co}}(u_p(z,t)) = \begin{cases} \frac{1}{60} 10^{\frac{u_p(z,t)-u_{\mathrm{ref}}}{\tilde{z}}}, & u_p(z,t) \ge u_{\mathrm{cutoff}} \\ 0, & \text{else} \end{cases} \tag{4.28}$$

for the product-specific z-value $\tilde{z}$ which depends on the reference temperature u_{ref} and a cut-off temperature u_{cutoff}.
The hyperbolic partial differential equations are given by

$$\frac{\partial u_c(z,t)}{\partial t} + v\frac{\partial u_c(z,t)}{\partial z} = -\kappa_1(u_c(z,t) - u_s(z,t)), \tag{4.29}$$

$$\frac{\partial u_p(z,t)}{\partial t} + v\frac{\partial u_p(z,t)}{\partial z} = -\kappa_2(u_p(z,t) - u_c(z,t)), \tag{4.30}$$

$$\frac{\partial \mathrm{PU}(z,t)}{\partial t} + v\frac{\partial \mathrm{PU}(z,t)}{\partial z} = -\mathrm{PU}_{\mathrm{co}}(u_p(z,t)). \tag{4.31}$$

Position and time become $z = z(\eta,\sigma)$ and $t = t(\eta,\sigma)$ when the coordinates η and σ are introduced.
In the case $t \le \frac{z}{v}$, the corresponding differential equations are given by

$$\begin{aligned} \frac{\partial u_c}{\partial \sigma} &= \frac{\partial u_c}{\partial z}\frac{\partial z}{\partial \sigma} + \frac{\partial u_c}{\partial t}\frac{\partial t}{\partial \sigma}, \\ \frac{\partial u_p}{\partial \sigma} &= \frac{\partial u_p}{\partial z}\frac{\partial z}{\partial \sigma} + \frac{\partial u_p}{\partial t}\frac{\partial t}{\partial \sigma}, \\ \frac{\partial \mathrm{PU}}{\partial \sigma} &= \frac{\partial \mathrm{PU}}{\partial z}\frac{\partial z}{\partial \sigma} + \frac{\partial \mathrm{PU}}{\partial t}\frac{\partial t}{\partial \sigma}. \end{aligned}$$

Comparison with the hyperbolic partial differential equations gives

$$\begin{aligned} &\frac{\partial t}{\partial \sigma} = 1, \ \frac{\partial z}{\partial \sigma} = v, \\ &\frac{\partial u_c}{\partial \sigma} = -\kappa_1(u_c(z(\eta,\sigma),t(\eta,\sigma)) - u_s(z(\eta,\sigma),t(\eta,\sigma))), \\ &\frac{\partial u_p}{\partial \sigma} = -\kappa_2(u_p(z(\eta,\sigma),t(\eta,\sigma)) - u_c(z(\eta,\sigma),t(\eta,\sigma))), \\ &\frac{\partial \mathrm{PU}}{\partial \sigma} = -\mathrm{PU}_{\mathrm{co}}(u_p(z(\eta,\sigma),t(\eta,\sigma))). \end{aligned}$$

With the initial conditions $u_c(z,0) = u_{c0}(z)$, $u_p(z,0) = u_{p0}(z)$, $\mathrm{PU}(z,0) = \mathrm{PU}_0(z)$, the solution of the system of ordinary differential equations is given by

$$
\begin{aligned}
&t(\eta,\sigma) = \sigma,\ z(\eta,\sigma) = \eta + v\sigma,\\
&u_c(\eta,\sigma) = u_c0(\eta)e^{-\kappa_1\sigma} + \kappa_1 \int_0^\sigma u_s(\eta + vs, s)e^{-\kappa_1(\sigma-s)}ds,\\
&u_p(\eta,\sigma) = u_p0(\eta)e^{-\kappa_2\sigma} + \kappa_2 \int_0^\sigma u_c(\eta + vs, s)e^{-\kappa_2(\sigma-s)}ds,\\
&\mathrm{PU}(\eta,\sigma) = \mathrm{PU}_0(\eta) + \int_0^\sigma \mathrm{PU}_{\mathrm{co}}(u_p(\eta + vs, s))ds.
\end{aligned}
$$

Transformation back into the (z,t)-plane gives $\sigma = t$ and $\eta = z - vt$. Therefore, the solution of the partial differential equation is given by

$$u_c(z,t) = u_c0(z - vt)e^{-\kappa_1 t} + \kappa_1 \int_0^t u_s(z - vt + vs, s)e^{-\kappa_1(t-s)}ds, \tag{4.32}$$

$$u_p(z,t) = u_p0(z - vt)e^{-\kappa_2 t} + \kappa_2 \int_0^t u_c(z - vt + vs, s)e^{-\kappa_2(t-s)}ds, \tag{4.33}$$

$$\mathrm{PU}(z,t) = \mathrm{PU}_0(z - vt) + \int_0^t \mathrm{PU}_{\mathrm{co}}(u_p(z - vt + vs, s))ds. \tag{4.34}$$

In the case $t > \frac{z}{v}$, the corresponding differential equations are given by

$$
\begin{aligned}
\frac{\partial u_c}{\partial \eta} &= \frac{\partial u_c}{\partial z}\frac{\partial z}{\partial \eta} + \frac{\partial u_c}{\partial t}\frac{\partial t}{\partial \eta},\\
\frac{\partial u_p}{\partial \eta} &= \frac{\partial u_p}{\partial z}\frac{\partial z}{\partial \eta} + \frac{\partial u_p}{\partial t}\frac{\partial t}{\partial \eta},\\
\frac{\partial \mathrm{PU}}{\partial \eta} &= \frac{\partial \mathrm{PU}}{\partial z}\frac{\partial z}{\partial \eta} + \frac{\partial \mathrm{PU}}{\partial t}\frac{\partial t}{\partial \eta}.
\end{aligned}
$$

Comparison with the hyperbolic partial differential equations gives

$$
\begin{aligned}
&\frac{\partial t}{\partial \eta} = 1,\ \frac{\partial z}{\partial \eta} = v,\\
&\frac{\partial u_c}{\partial \eta} = -\kappa_1(u_c(z(\eta,\sigma),t(\eta,\sigma)) - u_s(z(\eta,\sigma),t(\eta,\sigma))),\\
&\frac{\partial u_p}{\partial \eta} = -\kappa_2(u_p(z(\eta,\sigma),t(\eta,\sigma)) - u_c(z(\eta,\sigma),t(\eta,\sigma))),\\
&\frac{\partial t}{\partial \eta} = -\mathrm{PU}_{\mathrm{co}}(u_p(z(\eta,\sigma),t(\eta,\sigma))).
\end{aligned}
$$

With the boundary conditions $u_c(0,t) = u_{\text{in}}$, $u_p(0,t) = u_{\text{in}}$ and $\text{PU}(z,0) = 0$, the solution of the system of ordinary differential equations is given by

$$
\begin{aligned}
&t(\eta,\sigma) = \sigma + \eta,\ \ z(\eta,\sigma) = v\eta,\\
&u_c(\eta,\sigma) = u_{\text{in}} e^{-\kappa_1 \eta} + \kappa_1 \int_0^{\eta} u_s(vs,s) e^{-\kappa_1(\eta - s)} ds,\\
&u_p(\eta,\sigma) = u_{\text{in}} e^{-\kappa_2 \eta} + \kappa_2 \int_0^{\eta} u_c(vs,s) e^{-\kappa_2(\eta - s)} ds,\\
&\text{PU}(\eta,\sigma) = \int_0^{\eta} \text{PU}_{\text{co}}(u_p(vs,s)) ds.
\end{aligned}
$$

Transformation back into the (z,t)-plane gives $\sigma = t - \frac{z}{v}$ and $\eta = \frac{z}{v}$. Therefore, the solution of the partial differential equation is given by

$$u_c(z,t) = u_{\text{in}} e^{-\kappa_1 \frac{z}{v}} + \kappa_1 \int_0^{\frac{z}{v}} u_s(vs,s) e^{-\kappa_1(\frac{z}{v} - s)} ds, \tag{4.35}$$

$$u_p(z,t) = u_{\text{in}} e^{-\kappa_2 \frac{z}{v}} + \kappa_2 \int_0^{\frac{z}{v}} u_c(vs,s) e^{-\kappa_2(\frac{z}{v} - s)} ds, \tag{4.36}$$

$$\text{PU}(z,t) = \int_0^{\frac{z}{v}} \text{PU}_{\text{co}}(u_p(vs,s)) ds. \tag{4.37}$$

To simplify these equations, the system is discretized in time and space. It can be assumed that the spray water temperature $u_s(z,t)$ and the container temperature $u_c(z,t)$ are constant if the discretization is fine enough. Choosing the intervals for the time and space discretization as $\Delta t = \frac{\Delta z}{v}$, the solution at (z_n, t_m) can be used to compute the solution at (z_{n+1}, t_{m+1}). The approximated solution has small round-off errors if the intervals are parametrized along a characteristic line. The numerical solution is obtained by

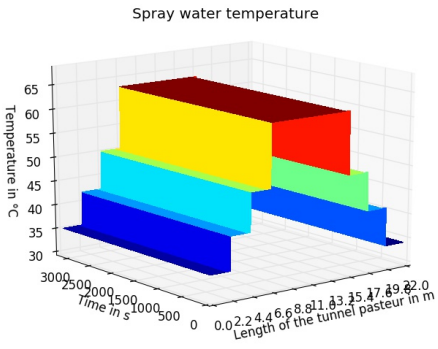
$$u_c(z_{n+1},t_{m+1}) = u_c(z_n,t_m) e^{-\kappa_1 \Delta t} + u_s(z_n,t_m)(1 - e^{-\kappa_1 \Delta t}), \tag{4.38}$$

$$u_p(z_{n+1},t_{m+1}) = u_p(z_n,t_m) e^{-\kappa_2 \Delta t} + u_c(z_n,t_m)(1 - e^{-\kappa_2 \Delta t}), \tag{4.39}$$

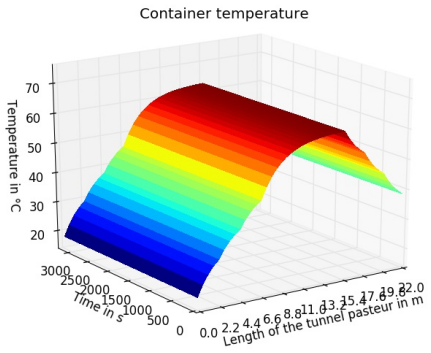
$$\text{PU}(z_{n+1},t_{m+1}) = \text{PU}(z_n,t_m) + \text{PU}_{\text{co}}(u_p(z_n,t_m)). \tag{4.40}$$

4.2.4 Example for Standard Pasteur

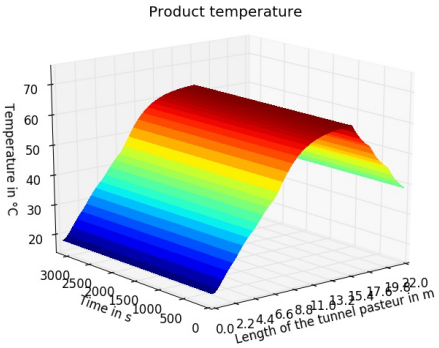
The results for constant spray water temperature are shown in Figure 4.2. In this case, it is sufficient to consider the static model, since the spray water temperature does not change with time and in consequence time and position can be coupled without loss of accuracy. This can be seen in Figure 4.2 by taking the curve of each plot in one time state t.



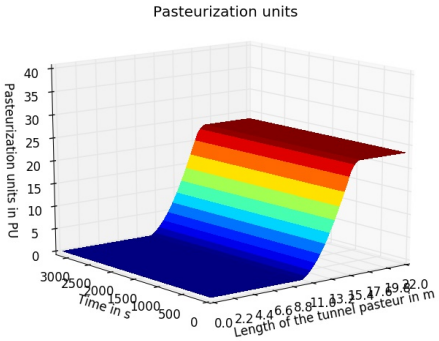
(a) Spray water temperature



(b) Container temperature



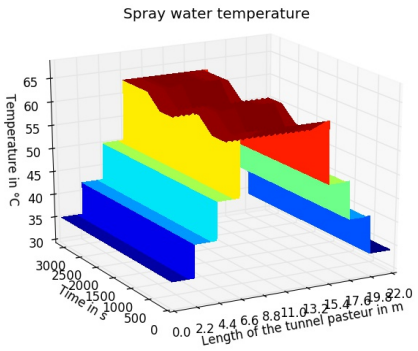
(c) Product temperature



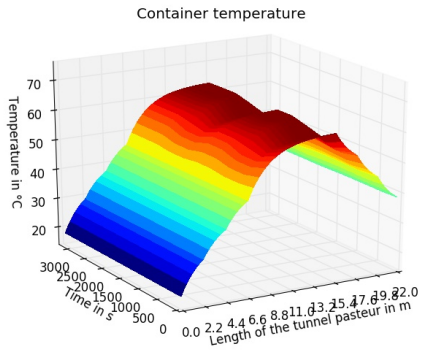
(d) Pasteurisation units

Figure 4.2: Solutions of the dynamic model assuming constant spray water temperature in time

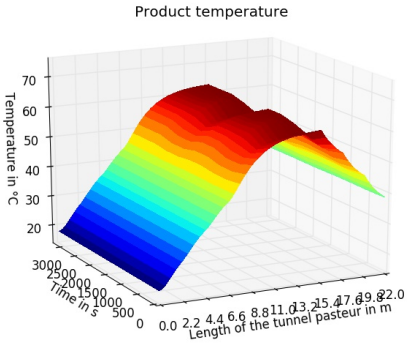
For a time varying spray water temperature, the results are shown in Figure 4.3. In this case, time and position can not be coupled. At every time state t, the spray water temperature function is different. This temperature function influences the container and thus the product temperature, which affects the amount of PUs in the product once the thermal treatment is finished.



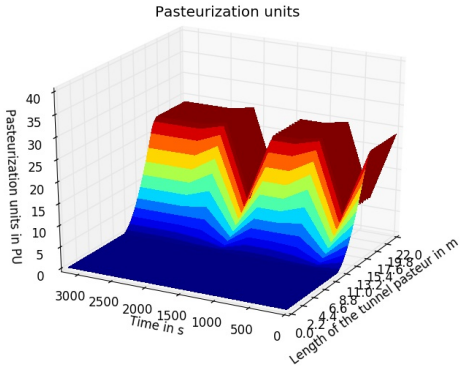
(a) Spray water temperature



(b) Container temperature



(c) Product temperature



(d) Pasteurisation units

Figure 4.3: Solutions of the dynamic model assuming non constant spray water temperature in time

Comparing Figure 4.2 and Figure 4.3, the dependence of the PUs on the spray water temperature becomes apparent.

5 Optimization

In this chapter, two optimization algorithms are presented. First a gradient-based algorithm is discussed, followed by a derivative-free. The choice of these two algorithms is motivated in Chapter 6 by implementational reasons as well as by requirements of the problem to be optimized. The main focus concentrates on the idea and the construction of the algorithms and their numerical implementation. The reader is referred to the literature for convergence properties. Basics about optimization problems are summarized in the next section and the implementation of the algorithms in the NLopt library in Python is outlined in the last section of this chapter.

5.1 Fundamentals About Optimization

Mathematically, optimization describes the problem to minimize or maximize a function subject to constraints on its variables. Let

- $x \in \mathbb{R}^n$ be the vector of variables, called unknowns,
- J be the objective or cost function, a scalar function of x to be optimized,
- c_i be the constraint functions, scalar functions of x defining certain equations and (in)equalities which the unknown vector x must satisfy.

The optimization problem is given by

$$\begin{aligned} \underset{x}{\text{minimize}}\ & J(x) \\ \text{subject to } & c_i(x) = 0,\ i \in \mathcal{E}, \\ & c_i(x) \leq 0,\ i \in \mathcal{I}. \end{aligned} \tag{5.1}$$

Here $\mathcal{E}$ and $\mathcal{I}$ are disjoint sets of indices for equality and inequality constraints. Points which satisfy all constraints are called feasible points. [14, p.2 ff.]

In optimization, the most difficult but also the most important step is the modeling process, more precisely to identify the objective function, the variables and the constraints. The model should neither be too simple, as it then would not give useful results about the problem, nor should it be too complex, as it might be unsolvable in this case.
Once the model is formulated, problem (5.1) can be minimized with different optimization algorithms. The choice of the algorithm is important as it determines how efficiently

the problem is solved, and whether a solution is found at all.

An optimization problem can be solved either locally or globally. A local minimum is a solution at which the objective function is smaller than all other feasible solutions in its neighborhood, whereas a global minimum is a solution with the lowest function value among all feasible solutions. Global solutions are often difficult to locate, as the optimization algorithm has to optimize the given objective function over the whole domain. Therefore, a local optimization is preferred.
The variables of the optimal value are computed iteratively by the optimization algorithm. Starting with an initial guess for the vector of variables x, a sequence of iterates approximating the solution is computed until a termination criterion is fulfilled. Optimization algorithms differ in the way they use one iterate to compute the next, but they have some properties in common. A good optimization method aims for robustness in the sense that it should be applicable succesfully to a large class of problems for all reasonable starting points. Moreover, it should be efficient in computational time and storage and provide a high accuracy both in the solution and the approximation process. [14, p.8 ff.]

Concepts frequently used in optimization are summarized here as far as this theory is used in the rest of the chapter. For proofs and further discussion, the reader is referred to the literature.

Lagrange Functions and KKT

Definition 5.1.1 (Lagrange function)**.** The Lagrange function of the optimization problem (5.1) is defined as

$$L(x;\lambda) = J(x) + \sum_{i\in\mathcal{I}} \lambda_i c_i(x) + \sum_{i\in\mathcal{E}} \lambda_i c_i(x) \tag{5.2}$$

with Lagrange multipliers $\lambda_i \geq 0$ for $i \in \mathcal{I}$ and λ_i free for $i \in \mathcal{E}$.

Definition 5.1.2 (KKT point)**.** A point $\bar{x} \in \mathbb{R}^n$ is called a KKT (Karush–Kuhn–Tucker) point, if there exist Lagrange multipliers λ such that

$$\begin{aligned}
&\nabla_x L(\bar{x};\lambda) = 0 \text{ (stationarity)},\\
&c_i(\bar{x}) \leq 0,\ i \in \mathcal{I} \text{ (primal feasibility)},\\
&c_i(\bar{x}) = 0,\ i \in \mathcal{E} \text{ (primal feasibility)},\\
&\lambda_i c_i(\bar{x}) = 0,\ i \in \mathcal{I} \text{ (complementary slackness)},\\
&\lambda_i \geq 0,\ i \in \mathcal{I} \text{ (dual feasibility)}.
\end{aligned} \tag{5.3}$$

For unconstrained convex optimization problems, the gradient is zero at a global minimum. The KKT conditions are the corresponding conditions for the global minimum of constrained convex optimization problems; in addition, they are not only a necessary, but even a sufficient condition.

Linear Programming and Duality

A linear program (LP) is a minimization problem given by a linear cost function,

$$\underset{x}{\text{minimize}}\, a^T x,\ x \in S \subseteq \mathbb{R}^n,$$

for a polyhedral set S. In order to study this LP problem with linear constraints, the following notation is introduced:

$$A = \begin{pmatrix} A_{J'} \\ A_{J''} \end{pmatrix} = \begin{pmatrix} A_{K'} & A_{K''} \end{pmatrix},\ b = \begin{pmatrix} b_{J'} \\ b_{J''} \end{pmatrix},\ c = \begin{pmatrix} c_{K'} \\ c_{K''} \end{pmatrix},\ x = \begin{pmatrix} x_{K'} \\ x_{K''} \end{pmatrix}$$

for J' and J'' disjunct sets in the row index set $\{1, \ldots, m\}$, and K' and K'' disjunct sets in the column index set $\{1, \ldots, n\}$ such that $x_k \geq 0$ for $k \in K'$ and free for $k \in K''$.

The *primal* LP problem is given by

$$\begin{aligned} &\underset{x}{\text{minimize}}\ a^T x \\ &\text{subject to } A_{J'} x \geq b_{J'},\ A_{J''} x = b_{J''},\ x_{K'} \geq 0,\ x_{K''} \text{ free.} \end{aligned} \tag{5.4}$$

The *dual* LP problem of (5.4) is given by

$$\begin{aligned} &\underset{y}{\text{maximize}}\ b^T y \\ &\text{subject to } A_{K'}^T y \leq a_{K'},\ A_{K''}^T y = b_{J''},\ y_{J'} \geq 0,\ y_{J''} \text{ free.} \end{aligned} \tag{5.5}$$

The solutions of (5.4) and (5.5) are connected to each other by the next theorem.

Theorem 5.1.1 (Complementary Slackness Principle)**.** The following conditions are equivalent for feasible $\hat{x}$ and $\hat{y}$:

(i) $\hat{x}$ and $\hat{y}$ are optimal for (5.4) and (5.5) respectively.

(ii) $a^T \hat{x} = b^T \hat{y}$.

(iii) $\hat{y}_j (b_j - (A\hat{x})_j) = 0$ for $j = 1 : m$, $\hat{x}_k (a_k - (A^T \hat{y})_k) = 0$ for $i = 1 : n$.

To construct the dual problem of a LP minimization problem allows to view this optimization problem from another perspective, which might simplify finding the optimal solution.

Trust Region

When constructing optimization methods, information about the cost function J can be used to construct a model function m_k, which approximates J in a neighbourhood of a given point $x^{(k)}$ and is aimed to be constructed such that its minimum can easily be computed. Since the function m_k approximates J only in a neighbourhood of $x^{(k)}$, the

search for an optimal solution of m_k should be restricted to the neighbourhood of $x^{(k)}$, called *trust region*. A candidate step p is found by solving the problem

$$\underset{p}{\text{minimize}}\ m_k(x^{(k)} + p) \tag{5.6}$$

for $x^{(k)} + p$ within the trust region. If the candidate solution does not give a sufficient decrease in the function value of J, the trust region is shrinked and problem (5.6) is solved again. The candidate step is usually set $\|p\|_2 \leq \Delta$ for the scalar trust region radius $\Delta > 0$.

5.2 Method of Moving Asymptotes

The *method of moving asymptotes* (MMA) is a class of the *conservative convex separable approximation* (CCSA) *methods*, described by Krister Svanberg in [16]. CCSA methods are designed to solve inequality-constrained nonlinear programming problems of the form

$$\begin{aligned} &\underset{x}{\text{minimize}}\ J(x) \\ &\text{subject to } c_i(x) \leq 0,\ i = 1:m \\ &\qquad\qquad x_j^{\min} \leq x_j \leq x_j^{\max},\ j = 1:n \end{aligned} \tag{5.7}$$

with $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ the vector of variables, $x_j^{\min}$ and $x_j^{\max}$ given real numbers and $J, c_1, \ldots, c_m$ real-valued, twice continuously differentiable functions.

Algorithms of CCSA method type approximate the original problem (5.7) by special convex problems which can be solved explicitly. The approximation routine is subdivided into inner and outer iterations. Outer iterations update the current iterate $x^{(k)}$ with a new iterate $x^{(k+1)}$ which is found with the inner iteration routine. Inner iterations take place within a given outer iteration $x^{(k)} \longrightarrow x^{(k+1)}$; the objective and constraint functions are approximated by certain convex separable functions around $x^{(k)}$ to be specified below, and the convex subproblem is solved. The optimal solution of the subproblem is either accepted or rejected. This choice is based on the following definition.

Definition 5.2.1 (Conservative function)**.** An approximating function is called conservative in a point, if its function value is greater than or equal to the original function value at this point.

If the approximating objective and constraint functions values at the optimal solution of the subproblem are conservative, the optimal solution of the subproblem is accepted and the outer iteration step is completed by setting $x^{(k+1)}$ to be the optimal solution of the subproblem. If the optimal solution of the subproblem is rejected, a new inner iteration with a modified subproblem is performed. This is repeated until an acceptable solution is found and the outer iteration step can be finished.
Finding conservative functions does not imply that the feasible set of the subproblem

is completely contained in the original feasible set, but the optimal solution of the subproblem is a feasible solution of the original problem with decreasing objective value for each iteration.
Each outer iterate requires function values and first order derivatives of the original objective and constraint functions, evaluated at the current iterate $x^{(k)}$.
Each inner iterate requires function values, but no derivatives, evaluated at the optimal solution of the most recent subproblem.

It is preferrable to transform problem (5.7) into a closely related problem with two vectors of variables.

$$\begin{aligned} \underset{x,y}{\text{minimize}}\ & J(x) + \sum_{i=1}^{m} \tilde{c}_i y_i \\ \text{subject to}\ & c_i(x) - y_i \le 0,\ i = 1:m \\ & c_j^{\min} \le x_j \le c_j^{\max},\ j = 1:n \\ & y_i \ge 0,\ i = 1:m \end{aligned} \tag{5.8}$$

The "natural" optimization parameter $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is completed by an "artificial" optimization parameter $y = (y_1, \ldots, y_m)^T \in \mathbb{R}^m$ and $\tilde{c}_i$ real positive numbers for $i = 1:m$. For very large $\tilde{c}_i$, in any optimal solution $(\hat{x}, \hat{y})$ of problem (5.8) one obtains $\hat{y} = 0$ and consequently $\hat{x}$ is an optimal solution of problem (5.7).
Several reasons motivate to use problem formulation (5.8) instead of (5.7). The existence of feasible solutions and of at least one optimal solution for problem (5.8) will be proved below. Moreover, each optimal solution of problem (5.8), either local or global, satisfies the KKT conditions which are related to those of problem (5.7).

Proposition 5.2.1. $\hat{x}$ is a KKT point of (5.7) if and only if $(\hat{x}, 0)$ is a KKT point of (5.8).

Proof. Let $\hat{x}$ be a KKT point of (5.7). Assume the corresponding Lagrange multipliers λ_i for the constraints $c_i(x) \le 0$ satisfy $\tilde{c}_i \ge \lambda_i$ for all i. Then $(x, y) = (\hat{x}, 0)$ is a KKT point of (5.8) with Lagrange multipliers λ_i for the constraints $c_i(x) - y_i \le 0$.
Let $(x, y) = (\hat{x}, 0)$ be a KKT point of problem (5.8) with Lagrange multiplier λ_i for the constraints $c_i(x) - y_i \le 0$ and $\lambda_i \le \tilde{c}_i$ for all i. Then $\hat{x}$ is a KKT point for problem (5.7) with Lagrange multipliers λ_i for the constraints $c_i(x) \le 0$. □

If there exists no KKT point for problem (5.7), then there exists no KKT point for problem (5.8) with $y = 0$, but there is always at least one KKT point for (5.8) with $y \ne 0$. This is proved by considering a further extended problem formulation.

$$\begin{aligned} \underset{x,y,z}{\text{minimize}}\ & J(x) + a_0 z + \sum_{i=1}^{m} (\tilde{c}_i y_i + \frac{1}{2} d_i y_i^2) \\ \text{subject to}\ & c_i(x) - a_i z - y_i \le 0,\ i = 1:m \\ & c_j^{\min} \le x_j \le c_j^{\max},\ j = 1:n \\ & z \ge 0,\ y_i \ge 0,\ i = 1:m \end{aligned} \tag{5.9}$$

with $a_0,\ a_i,\ \tilde{c}_i,\ d_i$ real numbers such that $a_0 > 0,\ a_i \geq 0,\ \tilde{c}_i \geq 0,\ d_i \geq 0$ and $\tilde{c}_i + d_i > 0$ for $i = 1 : m$ and $a_i\tilde{c}_i > a_0$ for all i such that $a_i > 0$. Moreover, $c_j^{\min}$ and $c_j^{\max}$ are real numbers such that $c_j^{\min} < c_j^{\max}$ for $j = 1 : n$. It will be proven below that problem (5.9) always has a feasible and at least one optimal solution, even if problem (5.7) does not have any feasible solution.
Setting $a_i = d_i = 0$ for $i = 1 : m$ and $a_0 = 1$ implies $z = 0$ in any optimal solution of problem (5.9). This shows that problem (5.8) is a special case of problem (5.9); in fact, both problem descriptions are equivalent. The subproblem which is constructed to approximate the optimal solution for (5.7) is of the form (5.9).

Problem (5.9) is equivalent to the typically nonsmooth problem (5.10) in the variables $(x_1, \ldots, x_n)^T \in \mathbb{R}^n$:

$$\begin{aligned} &\underset{x}{\text{minimize}}\ J(x) + a_0 \max_{i \in \mathcal{A}_1} \left\{ \frac{c_i^+(x)}{a_i} \right\} + \sum_{i \in \mathcal{A}_0} (\tilde{c}_i c_i^+(x) + \frac{1}{2} d_i (c_i^+(x))^2) \\ &\text{subject to } x \in X \end{aligned} \tag{5.10}$$

with

$$\begin{aligned} X &= \{x \in \mathbb{R}^n | c_j^{\min} \leq x_j \leq c_j^{\max},\ j = 1 : n\}, \\ \mathcal{A}_1 &= \{i \in \{1 : m\} | a_i > 0\}, \\ \mathcal{A}_0 &= \{i \in \{1 : m\} | a_i = 0\}, \\ c_i^+(x) &= \max\{0, c_i(x)\}. \end{aligned} \tag{5.11}$$

This form can be used to prove the existence of a global solution of problem (5.9).

Proposition 5.2.2. For fixed $x \in X$ in problem (5.9), the corresponding optimal values of the variables y and z are unique. If $\mathcal{A}_1 = \emptyset$, then $z = 0$ and $y_i = c_i^+(x)$ for $i = 1 : m$. If $\mathcal{A}_1 \neq \emptyset$, then $z = \max_{i \in \mathcal{A}_1} \{\frac{c_i^+(x)}{a_i}\}$ and $y_i = 0$ for $i \in \mathcal{A}_1$ and $y_i = c_i^+(x)$ for $i \in \mathcal{A}_0$.

Proof. If $\mathcal{A}_1 = \emptyset$, by the assumptions of problem (5.9) $a_0 > 0$ and $\tilde{c}_i + d_i > 0$ for $i = 1 : m$, implying $z = 0$ and $y_i = c_i^+(x)$.
If $\mathcal{A}_1 \neq \emptyset$, in addition the assumption $a_i\tilde{c}_i > a_0$ for $i \in \mathcal{A}_1$ holds. This implies $z = \max_{i \in \mathcal{A}_1} \{\frac{c_i^+(x)}{a_i}\}$ and $y_i = 0$ if $i \in \mathcal{A}_1$ and $y_i = c_i^+(x)$ if $i \in \mathcal{A}_0$. □

Since problem (5.9) is equivalent to problem (5.10), Proposition 5.2.2 directly shows the next result.

Proposition 5.2.3. $(\hat{x}, \hat{y}, \hat{z})$ is a global optimum of problem (5.9) if and only if $\hat{x}$ is a global optimum of problem (5.10) with $\hat{y}$ and $\hat{z}$ given by Proposition 5.2.2.

This is used to prove the following.

Proposition 5.2.4. There exists at least one global optimum of problem (5.9).

Proof. The objective function is continuous on the compact set X for problem (5.10). In consequence, there exists at least one global minimum for problem (5.10). By Proposition 5.2.3, there exists at least one global optimum for problem (5.9). □

5.2.1 Basic Structure of CCSA Methods

All CCSA methods have the same basic structure to update the approximation points:

A distinction is made between inner and outer iterations. In any outer iteration step, the objective and constraint functions are approximated by an element of a parameter-dependent family of convex functions.
In any inner iteration step, the parameters of this approximation are changed iteratively to fulfill certain conservativity conditions.

Let $k \geq 1$ be the index for the outer iteration and $\ell \geq 0$ the index for the inner iteration, then (k, ℓ) denotes the ℓth inner iteration within the kth outer iteration. For simplicity, the notation is changed from now on and $J(x) = c_0(x)$.
The starting point $(x^{(1)}, y^{(1)}, z^{(1)})$ is obtained by choosing an initial $x^{(1)} \in X$ and calculating $y^{(1)}$ and $z^{(1)}$ according to Proposition 5.2.2.
In each outer iteration, a subproblem is generated and solved to update $(x^{(k)}, y^{(k)}, z^{(k)})$ with $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)})$. To construct the subproblem, the set X is replaced by a certain convex subset $X^{(k)}$ and the functions $c_i(x)$ are approximated by certain strictly convex separable functions $g_i^{(k,0)}(x)$ such that $g_i^{(k,0)}(x^{(k)}) = c_i(x^{(k)})$ for $i = 0 : m$. The optimal solution of the subproblem is denoted by $(\hat{x}^{(k,0)}, \hat{y}^{(k,0)}, \hat{z}^{(k,0)})$.
If $g_i^{(k,0)}(\hat{x}^{(k,0)}) \geq c_i(\hat{x}^{(k,0)})$, the next iteration point becomes $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (\hat{x}^{(k,0)}, \hat{y}^{(k,0)}, \hat{z}^{(k,0)})$ and the current outer iteration step is completed without any inner iterations needed.
If an inner iteration is needed, a new subproblem at $x^{(k)}$ is generated and solved. The new approximating functions $g_i^{(k,1)}(x)$ must satisfy $g_i^{(k,1)}(x^{(k)}) = c_i(x^{(k)})$ for $i = 0 : m$ and be more conservative than $g_i^{(k,0)}(x)$ for those indices i for which the inequality above was violated. The optimal solution of the new subproblem is denoted by $(\hat{x}^{(k,1)}, \hat{y}^{(k,1)}, \hat{z}^{(k,1)})$.
If $g_i^{(k,1)}(\hat{x}^{(k,1)}) \geq c_i(\hat{x}^{(k,1)})$, the next iteration point becomes $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (\hat{x}^{(k,1)}, \hat{y}^{(k,1)}, \hat{z}^{(k,1)})$ and the outer iteration step is completed without any further inner iterations needed. Otherwise, an inner iteration is needed with new approximating functions $g_i^{(k,2)}(x)$.
The inner iterations are repeated until $g_i^{(k,\ell)}(\hat{x}^{(k,\ell)}) \geq c_i(\hat{x}^{(k,\ell)})$ for all $i = 0 : m$. This will happen after a finite number of iterations due to the construction of the subproblems. The next iteration point becomes $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (\hat{x}^{(k,\ell)}, \hat{y}^{(k,\ell)}, \hat{z}^{(k,\ell)})$ and the outer iteration step is completed with ℓ inner iterations needed.

A basic scheme of the algorithm can be found in Figure 5.5.

5.2.2 Approximating Functions

The CCSA subproblem for $k \geq 1$ and $\ell \geq 0$ is formulated as the modified optimization problem (5.9):

$$\begin{aligned} \underset{x,y,z}{\text{minimize}} \;\; & g_0^{(k,\ell)}(x) + a_0 z + \sum_{i=1}^{m} (\tilde{c}_i y_i + \frac{1}{2} d_i y_i^2) \\ \text{subject to} \;\; & g_i^{(k,\ell)}(x) - a_i z - y_i \leq 0, \;\; i = 1:m \\ & x \in X^{(k)}, \;\; y \geq 0, \;\; z \geq 0 \end{aligned} \tag{5.12}$$

with $X^{(k)} = X(x^{(k)}, \sigma^{(k)})$, and $\sigma^{(k)} = (\sigma_1^{(k)}, \ldots, \sigma_n^{(k)})^T$ a vector of strictly positive parameters. The convex set $X(\xi, \sigma) \subseteq X$ is defined by

$$X(\xi, \sigma) = [x \in X | x_j \in [\xi_j - 0.9\sigma_j, \xi_j + 0.9\sigma_j], \;\; j = 1:n]. \tag{5.13}$$

Each vector $\sigma^{(k)}$ belongs to a given compact set S,

$$S = \{\sigma \in \mathbb{R}^n | \sigma_j^{\min} \leq \sigma_j \leq \sigma_j^{\max}, \;\; j = 1:n\}, \tag{5.14}$$

with $\sigma_j^{\min}$ and $\sigma_j^{\max}$ real numbers such that $0 < \sigma_j^{\min} < \sigma_j^{\max} < \infty$.
The approximating functions $g_i^{(k,\ell)}(x)$ are formulated as

$$g_i^{(k,\ell)}(x) = v_i(x, x^{(k)}, \sigma^{(k)}) + \rho_i^{(k,\ell)} w_i(x, x^{(k)}, \sigma^{(k)}), \;\; i = 0:m. \tag{5.15}$$

$v_i(x, \xi, \sigma)$ and $w_i(x, \xi, \sigma)$ are real-valued functions to be specified below, defined on the set

$$D = [(x, \xi, \sigma) | \xi \in X, \;\; \sigma \in S, \;\; x \in X(\xi, \sigma)]. \tag{5.16}$$

The functions $g_i^{(k,\ell)}(x)$ need to satisfy the following conditions for $i = 0:m$ in order to ensure that they have suitable properties as approximating subproblems:

$$v_i(\cdot, \xi, \sigma), \;\; w_i(\cdot, \xi, \sigma) \in C^2(X(\xi, \sigma)), \tag{5.17a}$$
$$v_i(x, x, \sigma) = c_i(x) \;\; \forall x \in X, \tag{5.17b}$$
$$w_i(x, x, \sigma) = 0 \;\; \forall x \in X, \tag{5.17c}$$
$$\nabla_x v_i(x, x, \sigma) = \nabla_x c_i(x) \;\; \forall x \in X, \tag{5.17d}$$
$$\nabla_x w_i(x, x, \sigma) = 0 \;\; \forall x \in X, \tag{5.17e}$$
$$\nabla_{xx}^2 v_i(x, \xi, \sigma) \text{ positive semidefinite } \forall (x, \xi, \sigma) \in D, \tag{5.17f}$$
$$\nabla_{xx}^2 w_i(x, \xi, \sigma) \text{ positive definite } \forall (x, \xi, \sigma) \in D. \tag{5.17g}$$

The parameters $\rho_i^{(k,\ell)}$ are strictly positive; the larger $\rho_i^{(k,\ell)}$, the more conservative the approximation. Within a given outer iteration k, the only difference between two inner iterations are the values of $\rho_i^{(k,\ell)}$. By the conditions above it follows that $g_i^{(k,\ell)}$ are first order approximations of the original functions c_i at the current point, since properties (5.17b) and (5.17d) imply

$$g_i^{(k,\ell)}(x^{(k)}) = c_i(x^{(k)}) \text{ and } \nabla g_i^{(k,\ell)}(x^{(k)}) = \nabla c_i(x^{(k)}), \;\; i = 0:m.$$

Figure 5.1 shows first order convex approximations in the same point with increasing value $\rho_i^{(k,\ell)}$. For larger $\rho_i^{(k,\ell)}$, the function values of the approximation will increase and the approximation is more conservative, in consequence an acceptable solution can be found with higher probability.

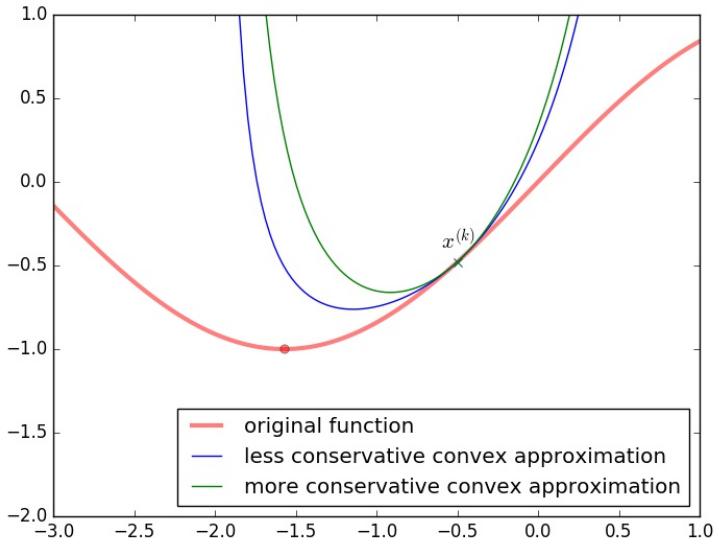


Figure 5.1: First order convex approximation in $x^{(k)} = -0.5$

Due to properties (5.17f) and (5.17g), the functions v_i and w_i are convex for $i = 0 : m$. Since $\rho_i^{(k,\ell)} > 0$, the functions $g_i^{(k,\ell)}$ are strictly convex. Moreover, they should be separable, more precisely

$$g_i^{(k,\ell)}(x) = g_{i0}^{(k,\ell)} + \sum_{j=1}^{n} g_{ij}^{(k,\ell)}(x_j).$$

For each fixed positive vector $\lambda \in \mathbb{R}^m$, the Lagrange function of the CCSA subproblem (5.12) is given by

$$L(x, y, z; \lambda) = g_0^{(k,\ell)}(x) + a_0 z + \sum_{i=1}^{m}(\tilde{c}_i y_i + \frac{1}{2} d_i y_i^2) + \sum_{i=1}^{m} \lambda_i (g_i^{(k,\ell)}(x) - a_i z - y_i)$$

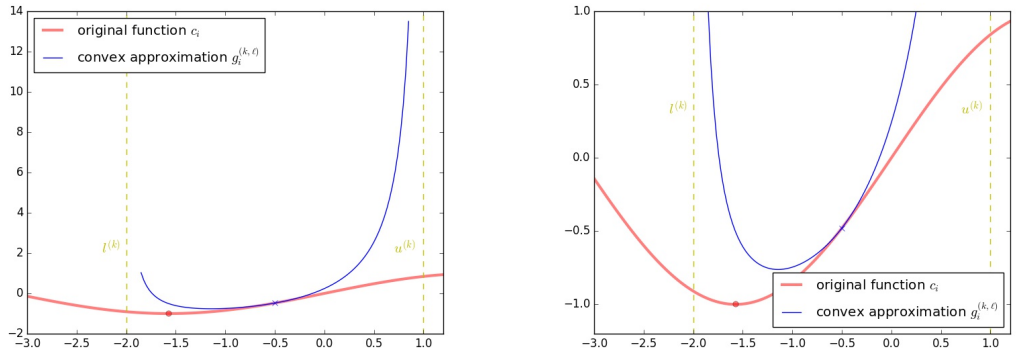
and can be minimized analytically with respect to $x \in X^{(k)}$, $y \geq 0$ and $z \geq 0$. If $d_i > 0$ for $i = 1 : m$ and a term εz^2 is added to the objective function, this analytical minimization gives a unique point $(\hat{x}(\lambda), \hat{y}(\lambda), \hat{z}(\lambda))$.
The concave dual function $\varphi(\lambda) = L(\hat{x}(\lambda), \hat{y}(\lambda), \hat{z}(\lambda), \lambda)$ becomes an explicit function, and the dual problem of maximizing $\varphi(\lambda)$ subject to the simple bounds $\lambda_i \geq 0$ for $i = 1 : m$ can be solved by conjugate gradient or a Newton-type method, taking into account the nonnegative constraints and the dual variables. If $\hat{\lambda}$ is an optimal solution of this dual problem, then $(x, y, z) = (\hat{x}(\hat{\lambda}), \hat{y}(\hat{\lambda}), \hat{z}(\hat{\lambda}))$ is the unique optimal solution of the CCSA problem.

5.2.3 MMA Approximations

The *method of moving asymptotes* is one strategy to determine the CCSA functions v_i and w_i. The approximating functions are chosen

$$g_i^{(k,\ell)}(x) = \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k,\ell)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k,\ell)}}{x_j - l_j^{(k)}} \right) + r_i^{(k,\ell)}. \tag{5.18}$$

The values $u_j^{(k)}$ and $l_j^{(k)}$ are upper and lower asymptotes, respectively, and are updated in each iteration, giving the method its name as shown in Figure 5.2.



(a) Convex approximation and moving asymptotes

(b) Convex approximation and moving asymptotes - zoomed in

Figure 5.2: Moving asymptotes: $l_j^{(k)}$ lower asymptote, $u_j^{(k)}$ upper asymptote

Svanberg presents in [16] a heuristic rule to to specify the moving asymptotes,

$$u_j^{(k)} = x_j^{(k)} + \sigma_j^{(k)} \text{ and } l_j^{(k)} = x_j^{(k)} - \sigma_j^{(k)}. \tag{5.19}$$

The coefficients $p_{ij}^{(k,\ell)}$, $q_{ij}^{(k,\ell)}$ and $r_i^{(k,\ell)}$ are set to be

$$p_{ij}^{(k,\ell)} = (\sigma_j^{(k)})^2 \max\{0, \frac{\partial c_i}{\partial x_j}(x^{(k)})\} + \frac{\rho_i^{(k,l)} \sigma_j^{(k)}}{4}, \tag{5.20}$$

$$q_{ij}^{(k,\ell)} = (\sigma_j^{(k)})^2 \max\{0, -\frac{\partial c_i}{\partial x_j}(x^{(k)})\} + \frac{\rho_i^{(k,l)} \sigma_j^{(k)}}{4}, \tag{5.21}$$

$$r_i^{(k,\ell)} = c_i(x^{(k)}) - \sum_{j=1}^{n} \frac{p_{ij}^{(k,\ell)} + q_{ij}^{(k,\ell)}}{\sigma_j^{(k)}}. \tag{5.22}$$

Since

$$g_i^{(k,\ell)}(x) = v_i(x, x^{(k)}, \sigma^{(k)}) + \rho_i^{(k,\ell)} w_i(x, x^{(k)}, \sigma^{(k)}),$$

it follows that

$$v_i(x,\xi,\sigma) = c_i(\xi) + \sum_{j=1}^{n} \frac{\sigma_j^2 \frac{\partial c_i}{\partial x_j}(\xi)(x_j - \xi_j) + \sigma_j |\frac{\partial c_i}{\partial x_j}(\xi)|(x_j - \xi_j)^2}{\sigma_j^2 - (x_j - \xi_j)^2}, \tag{5.23}$$

$$w_i(x,\xi,\sigma) = \frac{1}{2} \sum_{j=1}^{n} \frac{(x_j - \xi_j)^2}{\sigma_j^2 - (x_j - \xi_j)^2}. \tag{5.24}$$

Figure 5.3 shows examples of MMA approximations in different points and fixed values σ and ρ.

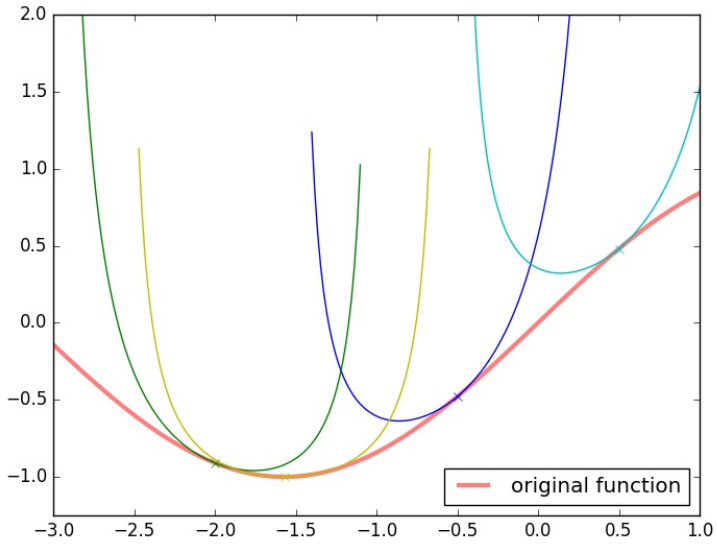


Figure 5.3: Convex approximations in different points for fixed σ and ρ

5.2.4 Rules for Updating $\rho_i^{(k,\ell)}$ and $\sigma_j^{(k)}$

The values $\rho_i^{(k,\ell)}$ and $\sigma_j^{(k)}$ influence the curvature of the convex approximation in the iteration point $x^{(k)}$ and in consequence its conservativeness. Thus, updating these values depends on the behaviour of the iteration process, specifically whether it is monotone or oscillating.

For $\ell = 0$ and a fixed, strictly positive small number $\rho_i^{\min}$ (i.e. 10^{-5}), it is advisable to set

$$\begin{aligned} \rho_i^{(1,0)} &= 1, \\ \rho_i^{(k+1,0)} &= \max\{0.1\rho_i^{(k,\hat{\ell}(k))}, \rho_i^{\min}\} \end{aligned} \tag{5.25}$$

with $\hat{\ell}(k)$ the number of inner iterations within the kth outer iteration such that $\rho_i^{(k,\hat{\ell}(k))}$ is the latest value of $\rho_i^{(k,\ell)}$.
In each inner iteration, updating $\rho_i^{(k,\ell)}$ is based on the solution of the most recent subproblem. The solution is rejected if $g_i^{(k,\ell)}(\hat{x}^{(k,\ell)}) < c_i^{(k,\ell)}(\hat{x}^{(k,\ell)})$ and a correction value $\rho_i^{(k,\ell+1)}$ must be chosen such that $g_i^{(k,\ell+1)}(\hat{x}^{(k,\ell)}) \geq c_i^{(k,\ell)}(\hat{x}^{(k,\ell)})$. This is achieved for $\rho_i^{(k,\ell+1)} = \rho_i^{(k,\ell)} + \delta_i^{(k,\ell)}$ with

$$\delta_i^{(k,\ell)} = \frac{c_i(\hat{x}^{(k,\ell)}) - g_i^{(k,\ell)}(\hat{x}^{(k,\ell)})}{w_i(\hat{x}^{(k,\ell)}, x^{(k)}, \sigma^{(k)})}. \tag{5.26}$$

Some modification of these values are required to get a globally convergent method:

$$\rho_i^{(k,\ell+1)} = \begin{cases} \min\{10\rho_i^{(k,\ell)}, 1.1(\rho_i^{(k,\ell)} + \delta_i^{(k,\ell)})\}, & \delta_i^{(k,\ell)} > 0, \\ \rho_i^{(k,\ell)}, & \delta_i^{(k,\ell)} \leq 0. \end{cases} \tag{5.27}$$

At the beginning of each inner iteration, the value $\rho_i^{(k,\ell)}$ is increased or unaltered, but never decreased. In order to avoid a too conservative method, it is important to be able to increase these values again in the begin of each new outer iteration.

Updating the value $\sigma_j^{(k)}$ depends on the functions v_i and w_i. For the MMA method, the Hessian matrix $\nabla_{xx}^2 w_i(x,\xi,\sigma)$ is diagonal with $\frac{\partial^2 w_i}{\partial^2 x_j^2}(x,\xi,\sigma) = \frac{\sigma_j^4}{(\sigma_j^2-(x_j-\xi_j)^2)^3} \geq \frac{\sigma_j^4}{\sigma_j^6} = \frac{1}{\sigma_j^2}$ for $j = 1:n$ and $(x,\xi,\sigma) \in D$ with equality if $x_j = \xi_j$. The curvature of the function w_i is in the direction of x_j, in consequence it increases with decreasing values of σ_j. The heuristic rule presented by Svanberg in [16] suggest to stabilize an oscillating x_j by a decreasing value of the corresponding σ_j. If x_j is monotonically increasing (or decreasing), it should be raised by an increased value of the corresponding σ_j. A possible way to achive this is to set

$$\sigma_j^{(k)} = \begin{cases} 0.5(x_j^{\max} - x_j^{\min}), & k = 1,2, \\ \gamma_j^{(k)}\sigma_j^{(k-1)}, & k \geq 3, \end{cases} \tag{5.28}$$

with

$$\gamma_j^{(k)} = \begin{cases} 0.7, & (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) < 0, \\ 1.2, & (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) > 0, \\ 1, & (x_j^{(k)} - x_j^{(k-1)})(x_j^{(k-1)} - x_j^{(k-2)}) = 0, \end{cases} \tag{5.29}$$

provided this leads to values satisfying

$$0.01(x_j^{\max} - x_j^{\min}) \leq \sigma_j^{(k)} \leq 10(x_j^{\max} - x_j^{\min}). \tag{5.30}$$

If any of these bounds is violated, $\sigma_j^{(k)}$ is set to the violated bound. Therefore,

$$\sigma_j^{\min} = 0.01(x_j^{\max} - x_j^{\min}),\ \sigma_j^{\max} = 10(x_j^{\max} - x_j^{\min}) \tag{5.31}$$

Figure 5.4 shows a few exemplary iterations applying the MMA routine.

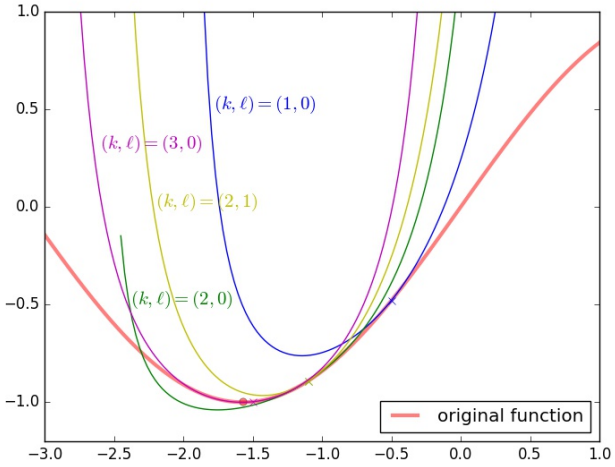


Figure 5.4: MMA routine in the one-dimensional case

5.2.5 Basic Scheme of the Algorithm

Figure 5.5 shows the basic iteration scheme of the MMA algorithm.

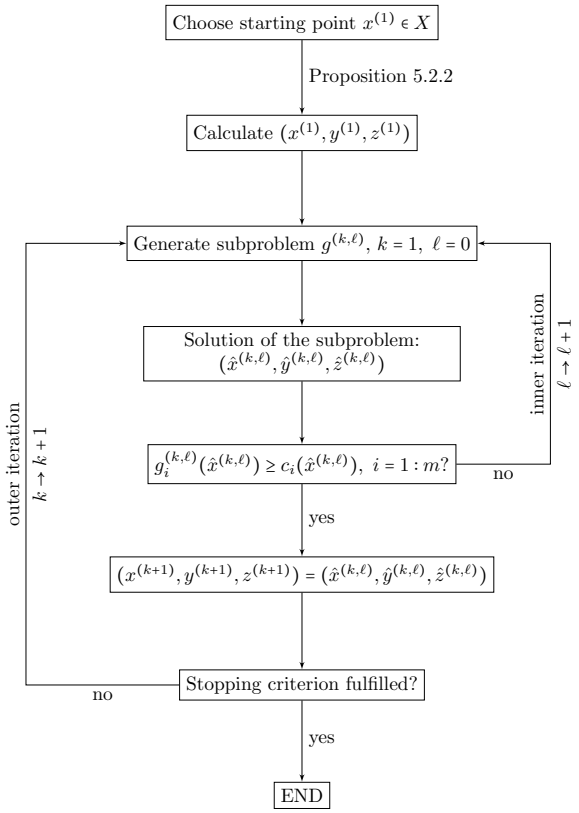


Figure 5.5: Summary of the MMA algorithm

5.3 Constrained Optimization by Linear Approximation

M.J.D. Powell describes in [15] a gradient-free algorithm to solve optimization problems based on *constrained optimization by linear approximation* (COBYLA).
The optimization problem to be solved is of the following form:

$$\begin{aligned} &\underset{x}{\text{minimize}} \quad J(x) \\ &\text{subject to } c_i(x) \geq 0, \ i = 1:m \end{aligned} \tag{5.32}$$

with $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ the vector of variables and $J, c_1, \ldots, c_m$ real-valued functions.[1]

The iterative linear approximation of the optimal solution of (5.32) is based on the method of Nelder and Mead (1965). This method aims to calculate the least value of a given function $F(x)$ without constraints on the variables $x \in \mathbb{R}^n$, described in [9, p.99 ff]. A simplex in $\mathbb{R}^n$ is a set of $n+1$ corner points $x^{(0)}, \ldots, x^{(n)}$ such that the vectors $x^{(j)} - x^{(0)}$ are linearly independent for $j = 1:n$. For a given simplex, $x^{(\ell)}$ denotes the vertex at which the objective function is worst, i.e. largest,

$$F(x^{(\ell)}) = \max\{F(x^{(j)}) | j = 0:n\}$$

and $x^{(m)}$ the vertex at which the objective function is best, i.e. smallest,

$$F(x^{(m)}) = \min\{F(x^{(j)}) | j = 0:n\}.$$

In each iteration step, the worst vertex $x^{(\ell)}$ in the simplex is replaced by a better one $x^{(\ell)}_{\text{new}}$ in order to converge to the minimum of F.
The reflection of $x^{(\ell)}$ in the face of the simplex defined by the other vertices is given by

$$x^{(r)} = -\alpha x^{(\ell)} + \frac{1+\alpha}{n} \sum_{j=0, j\neq\ell}^{n} x^{(j)} \tag{5.33}$$

for $\alpha > 0$. There exist three cases to set $x^{(\ell)}_{\text{new}}$, visualized in Figure 5.6 for the two-dimensional case.

Case 1: $F(x^{(m)}) < F(x^{(r)}) < F(x^{(\ell)})$
Set $x^{(\ell)}_{\text{new}} = x^{(r)}$.

Case 2: $F(x^{(r)}) \leq F(x^{(m)})$
A good direction is found in which to move the vertex, thus the simplex is expanded in this direction:

$$x^{(e)} = \beta x^{(r)} + \frac{1-\beta}{n} \sum_{j=0, j\neq\ell}^{n} x^{(j)} \tag{5.34}$$

for $\beta > 1$. Set

$$x^{(\ell)}_{\text{new}} = \begin{cases} x^{(e)}, & F(x^{(e)}) < F(x^{(r)}) \\ x^{(r)}, & F(x^{(e)}) \geq F(x^{(r)}). \end{cases}$$

[1] The formulation of the optimization problem is based on the original description of the COBYLA algorithm in [15]

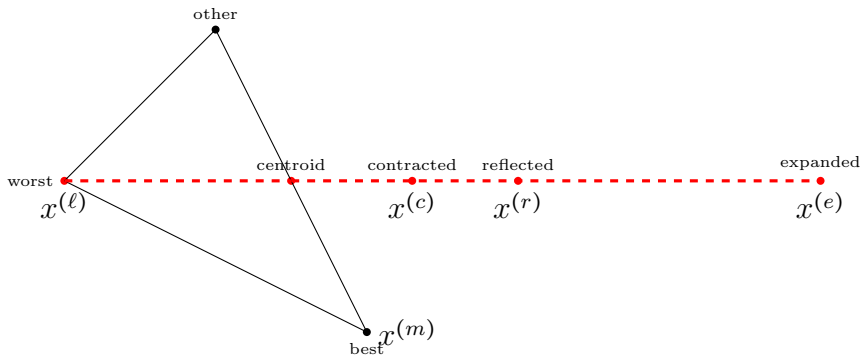


Figure 5.6: Three candidates to replace the worst vertex in the Nelder and Mead method in $\mathbb{R}^2$

Case 3: $F(x^{(r)}) \geq F(x^{(\ell)})$
A bad direction is found. Thus, the simplex is contracted by setting

$$x^{(c)} = \gamma x^{(r)} + \frac{1-\gamma}{n} \sum_{j=0, j\neq\ell}^{n} x^{(j)}$$

for $0 < \gamma < 1$. If $F(x^{(c)}) < F(x^{(\ell)})$, set $x_{\text{new}}^{(\ell)} = x^{(c)}$. Otherwise, the simplex is shrinked by putting $x^{(j)} = \frac{1}{2}(x^{(j)} + x^{(\ell)})$ for $j = 0 : n$ and restarting the iteration. The shrinking is skeched in Figure 5.7.

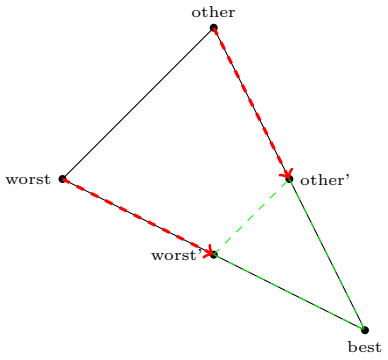


Figure 5.7: Shrinking the simplex in the Nelder and Mead method in $\mathbb{R}^2$

An example for the Nelder and Mead method in the two dimensional case is given in Figure 5.8.

Inspired by the method of Nelder and Mead, the vector of variables in the COBYLA method is iteratively generated from function values at the vertices $\{x^{(j)} | j = 0 : n\}$ of a nondegenerate simplex in $\mathbb{R}^n$. This allows to find unique linear functions $\hat{J}$ and $\hat{c}_i$ for

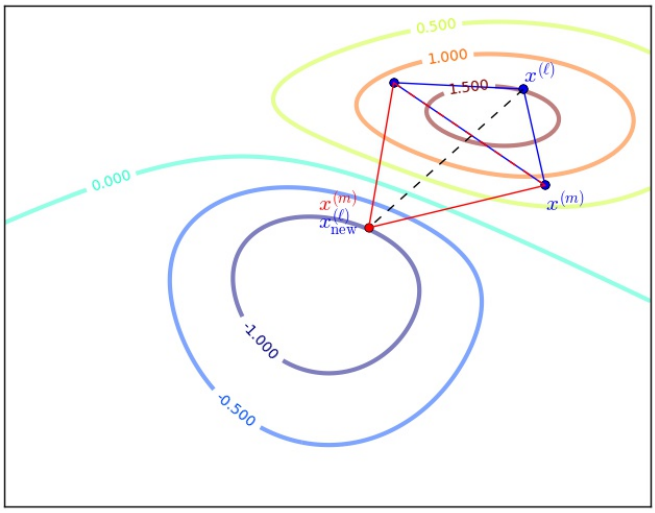


Figure 5.8: Example for optimization with the Nelder and Mead method in $\mathbb{R}^2$

$i = 1 : m$ which interpolate J and c_i at the $n+1$ vertices. Problem (5.32) is approximated by the linear programming problem

$$\begin{aligned} &\text{minimize } \hat{J}(x) \\ &\text{subject to } \hat{c}_i(x) \geq 0, \ i = 1 : m \end{aligned} \tag{5.35}$$

with $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ the vector of variables and $\hat{J}, \hat{c}_1, \ldots, \hat{c}_m$ linear real-valued functions.

By a trust region bound, changes in the variables are restricted. The trust region radius remains constant until predicted improvements to the objective function and feasible conditions fail to occur. In this case, the trust region radius is reduced until it reaches a minimal value defined beforehand. In order to avoid errors in the first iterations, the length of the trial steps should equal the current trust region bound. A function to compare the goodness of different vectors of variables is defined by

$$\Phi(x) = J(x) + \mu \max\{0, \max\{-c_i(x) | i = 1 : m\}\} \tag{5.36}$$

for $x \in \mathbb{R}^n$ and $\mu \in \mathbb{R}$. The function Φ is known as *auxiliary function* in the penalty function method in constrained optimization. For any feasible $\bar{x}$ it follows $\Phi(\bar{x}) = J(\bar{x})$. The vector $x \in \mathbb{R}^n$ is defined to be better than $y \in \mathbb{R}^n$ if and only if $\Phi(x) \leq \Phi(y)$.

5.3.1 Basic Structure of COBYLA

Let $x \in \mathbb{R}^n$ be the vector of variables calculated from the LP problem (5.35). This calculation requires the vertices $\{x^{(j)} | j = 0 : n\}$ of a nondegenerate simplex, a positive

trust region radius ρ and the current value of the parameter μ for the auxiliary function (5.36). The vertices are supposed to be ordered such that $x^{(0)}$ is the optimal vertex, i.e.

$$\Phi(x^{(0)}) \leq \Phi(x^{(j)}),\ j = 1:n. \tag{5.37}$$

The trust region condition on the new vector of variables $x^{(*)}$ is the bound

$$\|x^{(*)} - x^{(0)}\|_2 \leq \rho. \tag{5.38}$$

The vector $x^{(*)}$ should minimize the linear approximation $\hat{J}(x^{(*)})$ subject to (5.38) and the linear constraints

$$\hat{c}_i(x^{(*)}) \geq 0,\ i = 1:m. \tag{5.39}$$

If several minimal $x^{(*)}$ are found, the vector giving the least value of $\|x^{(*)} - x^{(0)}\|_2$ is chosen. It might happen that conditions (5.38) and (5.39) contradict each other. In this case, $x^{(*)}$ is defined by minimizing the greatest of the constraint violations $\{-\hat{c}_i(x^{(*)})|i = 1:m\}$ subject to the trust region bound (5.38). Any remaining freedom in $x^{(*)}$ is used to minimize $\hat{J}(x^{(*)})$ and if some freedom still remains, $\|x^{(*)} - x^{(0)}\|_2$ is made as small as possible. This process of finding the optimal x^* can be considered as an inner iteration like in the MMA algorithm.

5.3.2 Rules for Updating μ and ρ

In the auxiliary function (5.36), μ is initially set zero. To choose the optimal vertex, it is assumed that μ is a tiny positive number without further specification. The reduction $\Phi(x^{(*)}) < \Phi(x^{(0)})$ can not be expected if μ does not provide $\hat{\Phi}(x^{(*)}) < \hat{\Phi}(x^{(0)})$, with the auxiliary function for the linear approximation problem:

$$\hat{\Phi}(x) = \hat{J}(x) + \mu \max\{0, \max\{-\hat{c}_i(x)|i = 1:m\}\}. \tag{5.40}$$

Let $\bar{\mu}$ be the smallest nonnegative value of μ which would yield $\hat{\Phi}(x^{(*)}) < \hat{\Phi}(x^{(0)})$. If $\mu \geq \frac{3}{2}\bar{\mu}$, it is left unchanged, otherwise μ is iscreased to $2\bar{\mu}$. Increasing μ might result in violating condition (5.37). The optimality of $x^{(0)}$ can be preserved by exchanging two vertices of the simplex. The calculation of $x^{(*)}$ and any further adjustments on μ are repeated until $x^{(0)}$ is optimal and μ acceptable. The procedure will not cycle since each change to $x^{(0)}$ must reduce the value of $\max\{0, \max\{-c_i(x^{(0)})|i = 1:m\}\}$.

The process of modifying the trust region radius is likewise inspired by the Nelder and Mead method. The current trust region radius ρ is used until the iterations fail to provide satisfactory reductions in Equation (5.36); in this case, ρ should be decreased. The simplex can have different possible shapes which might cause the LP problem (5.35) to be a very poor approximation of the original problem (5.32). Thus, it is necessary to have an acceptable simplex before decreasing ρ. The trust region radius needs to be reduced if either $\|x^{(*)} - x^{(0)}\|_2 < \frac{1}{2}\rho$ or

$$\Phi(x^{(0)}) - \Phi(x^{(*)}) < 0.1(\hat{\Phi}(x^{(0)}) - \hat{\Phi}(x^{(*)})), \tag{5.41}$$

implying that changing the variables from $x^{(0)}$ to $x^{(*)}$ fails to provide a tenth of the improvement in Equation (5.40).
The initial and the final trust region radii ρ_{beg} and ρ_{end} are set beforehand; it is recommendable to choose ρ_{end} being approximately the required distance from the final vector of variables to the solution of the optimization problem, i.e. the error tolerance.
If $\rho \leq \rho_{\text{end}}$, the algorithm is terminated and the final vector of variables is the current $x^{(0)}$. The vector $x^{(*)}$ is preferred if $\Phi(x^{(*)})$ is available and satisfies $\Phi(x^{(*)}) < \Phi(x^{(0)})$.
If $\rho > \rho_{\text{end}}$,

$$\rho_{\text{new}} = \begin{cases} \frac{1}{2}\rho, & \rho > 3\rho_{\text{end}} \\ \rho_{\text{end}}, & \rho \leq 3\rho_{\text{end}} \end{cases} \tag{5.42}$$

It needs to be estimated whether it is advantageous to decrease μ when ρ is reduced. The ith constraint is important to the auxiliary function (5.36), if it is in the set

$$\mathcal{I} = \left\{ i | c_i^{(\min)} < \frac{1}{2} c_i^{(\max)} \right\} \cap \{1 : m\},$$

with $c_i^{(\min)}$, $c_i^{(\max)}$ the smallest and the largest values of c_i at the vertices of the current simplex. μ is set to be zero if $\mathcal{I} = \emptyset$, otherwise

$$\mu = \frac{\max_{j=0:n} J(x^{(j)}) - \min_{j=0:n} J(x^{(j)})}{\min\{\max[0, c_i^{(\max)}] - c_i^{(\min)} | i \in \mathcal{I}\}},$$

provided this change reduces μ. In contrast to the Nelder and Mead method, the simplex is preserved when ρ is decreased in the COBYLA algorithm. Any change to $x^{(*)}$ is caused by the new right hand side of Inequality (5.38). Moreover, the current simplex will be modified if it is not acceptable for the new value of ρ. Let $\sigma^{(j)}$ describe the Euclidean distance from the vertex $x^{(j)}$ to the opposite face of the current simplex, $\eta^{(j)}$ the length of the edge between $x^{(j)}$ and $x^{(0)}$ for $j = 1 : n$. A simplex is called acceptable if and only if

$$\begin{cases} \sigma^{(j)} \geq \alpha\rho, \\ \eta^{(j)} \leq \beta\rho, \end{cases} \quad j = 1 : n \tag{5.43}$$

with $0 < \alpha < 1 < \beta$ constants (in the code $\alpha = \frac{1}{4}$, $\beta = 2.1$). This implies the lengths of the edges and the volume of an acceptable simplex are ρ and ρ^n respectively.

5.3.3 Rules for Updating the Simplex

The first simplex is constructed with the initial vector of variables $x^{(0)}$ provided by the user and $x^{(j)} = x^{(0)} + \rho_{\text{beg}} e_j$, with $e_j \in \mathbb{R}^n$ the jth standard basis vector for $j = 1 : n$. $x^{(j)}$ is exchanged with $x^{(0)}$ if and only if $J(x^{(j)}) < J(x^{(0)})$. In this case, $x^{(0)}$ is the optimal vertex of the initial simplex. $x^{(*)}$ is not calculated in every iteration, instead $x^{(\Delta)}$ is calculated, which will be defined below. $x^{(*)}$ is calculated if and only if at least one of the following conditions holds:

i) There is no previous iteration.

ii) The previous iteration reduced ρ.

iii) The previous iteration calculated $x^{(\Delta)}$.

iv) The previous iteration calculated $x^{(*)}$ and reduced Equation (5.36) by at least one tenth of the predicted reduction.

v) The current simplex is acceptable.

If none of these condition holds, $x^{(\Delta)}$ is computed. If any of $\{\eta^{(j)}|j = 1 : n\}$ defined in Inequalities (5.43) is greater than $\rho\beta$, let ℓ be the least integer in $[1, n]$ satisfying

$$\eta^{(\ell)} = \max\{\eta^j|j = 1, : n\}.$$

Otherwise, ℓ is chosen such that

$$\sigma^{(\ell)} = \min\{\sigma^j|j = 1 : n\},$$

with $\sigma^{(\ell)} < \alpha\rho$, which is implied by the failure of condition v). The iteration replaces the vertex $x^{(\ell)}$ by $x^{(\Delta)}$, so $x^{(\Delta)}$ is required to be away from the face of the simplex opposite to the vertex $x^{(\ell)}$. Let $v^{(\ell)}$ be the vector of unit length orthogonal to this face, and

$$x^{(\Delta)} = x^{(0)} \pm \gamma\rho v^{(\ell)},$$

with the $\pm$ sign chosen to minimize $\hat{\Phi}(x^{(\Delta)})$ and a constant $\gamma \in (\alpha, 1)$ (in the code $\gamma = 0.5$). The next iteration is given by the simplex with vertices $\{x^{(j)}|j = 0 : n,\ j \neq \ell\}$ and $x^{(\Delta)}$.
If an iteration forms $x^{(*)}$, it must be chosen between three options:

- Reducing ρ,
- Preserving ρ for another iteration which calculates $x^{(*)}$,
- Preserving ρ for another iteration that gives priority to improve the acceptability of the simplex.

Since Inequality (5.41) is tested if and only if $x^{(*)}$ satisfies

$$\|x^{(*)} - x^{(0)}\|_2 \geq \frac{1}{2}\rho, \tag{5.44}$$

$J(x^{(*)})$ and $\{c_i(x^{(*)})|i = 1 : m\}$ are calculated only if (5.44) holds. These function values are included in future linear approximations by replacing one of the vertices $\{x^{(j)}|j = 1 : n\}$ of the current simplex with $x^{(*)}$. Let $\bar{\sigma}^{(j)}$ denote the distance from $x^{(*)}$ to the face of the current simplex being opposite $x^{(j)}$. Some elementary geometry shows that the volume of the simplex is multiplied by the factor $\frac{\bar{\sigma^{(j)}}}{\sigma^{(j)}}$ if $x^{(j)}$ is replaced by $x^{(*)}$. Thus, the nonsingularity of the interpolation is not damaged if j is in the set

$$\mathcal{J} = \{j|\bar{\sigma}^{(j)} \geq \sigma^{(j)}\} \cup \{j|\bar{\sigma}^{(j)} \geq \alpha\rho\}.$$

The optimal vertex of the next iteration is given by

$$\bar{x}^{(0)} = \begin{cases} x^{(*)}, & \Phi\big(x^{(*)}\big) < \Phi\big(x^{(0)}\big) \\ x^{(0)}, & \Phi\big(x^{(*)}\big) \geq \Phi\big(x^{(0)}\big). \end{cases} \tag{5.45}$$

If $\mathcal{J}$ is nonempty, let ℓ be the least element of $\mathcal{J}$ having the property

$$\|x^{(\ell)} - x^{(0)}\|_2 = \max\{\|x^{(j)} - x^{(0)}\|_2 | j \in \mathcal{J}\}.$$

The algorithm gives attention to the second of the acceptability requirements by replacing $x^{(\ell)}$ with $x^{(*)}$ if $\|x^{(\ell)} - x^{(0)}\|_2 > \delta\rho$ for $1 < \delta \leq \beta$ (in the code $\delta = 1.1$).
Otherwise, a new simplex is determined, with the requirement that its volume has to be maximized subject to the condition that updating is mandatory when $\Phi\big(x^{(*)}\big) < \Phi\big(x^{(0)}\big)$ and $\bar{\sigma}^{(\ell)} > \sigma^{(\ell)}$ holds. In this case,

$$\frac{\bar{\sigma}^{(\ell)}}{\sigma^{(\ell)}} = \max\left\{\frac{\bar{\sigma}^{(j)}}{\sigma^{(j)}} | j = 1:n\right\}.$$

The simplex is revised by most iterations, only if in addition to all inequalities $\{\bar{\sigma}^{(j)} \leq \sigma^{(j)} | j = 1:n\}$ and $\Phi\big(x^{(*)}\big) \geq \Phi\big(x^{(0)}\big)$ holds, then $\|x^{(j)} - x^{(0)}\|_2$ is bounded from above by $\delta\rho$ for all $j \in \mathcal{J}$.

5.3.4 Basic Scheme of the Algorithm

Figure 5.9 shows the basic scheme of the COBYLA algorithm.

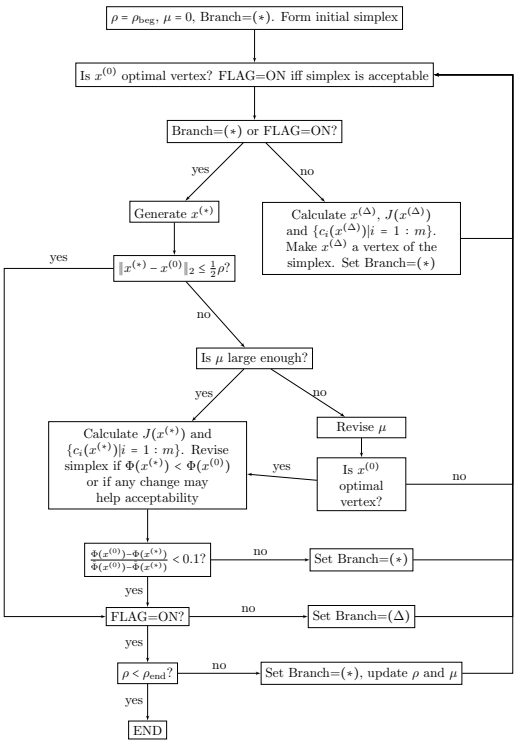


Figure 5.9: Summary of the COBYLA algorithm

5.4 Implementation in NLopt

Both algorithms introduced above are implemented in the open-source NLopt library for nonlinear optimization. Callable from C, C++, Fortran, Matlab, Python and other programming languages, NLopt provides a common interface for different optimization routines. The implemented algorithms support local and global, derivative-free and gradient-based as well as unconstrained and constrained optimization routines [2]. This section serves as an introduction to the implementation of optimization problems in NLopt using Python.

5.4.1 Objective Function and Constraints

The NLopt class in Python is initialized by setting the objective function and the constraints as outlined below.
The objective function takes NumPy arrays as arguments for the vector of variables and the gradient.

```
def costfunc(x,grad):
    if grad.size > 0:
    #array containing the partial derivatives
    return #value of costfunc(x)
```

`x` is a vector of length n and specifies the unknowns of the optimization problem. The gradient `grad` is an array of length n if a gradient-based optimization algorithm is used, and an empty array in the case of a derivative-free algorithm. This input is automatically generated by the chosen optimization routine. The constraints are defined in a similar way.

```
def constraint(x,grad):
    if grad.size > 0:
    #array containing the partial derivatives
    return #value of constraint(x)
```

The return value of the constraint function must be smaller than or equal to zero. The optimization algorithm is specified by calling

```
opt = nlopt.opt(algorithm,n)
```

where the algorithm and the dimension of the problem have to be determined. In Python, the `algorithm` values are of the form `nlopt.{G,L}{N,D}_XXXX` where `G/L` denotes if the optimization is global or local and `N/D` if the algorithm is derivative-free or gradient-based. `XXXX` specifies the used algorithm. Implemented algorithms and the names for calling them can be found in the NLopt Python Reference [3]. The dimension of the problem must be the length of the vector `x`. To specify if the objective function should be minimized or maximized, the methods

```
opt.set_min_objective(costfunc)
opt.set_max_objective(costfunc)
```

need to be called. If supported by the algorithm, bound constraints can be defined by

```
opt.set_lower_bounds(lb)
opt.set_upper_bounds(ub)
```

with `lb` and `ub` arrays of length n determining the lower and upper bound, repsectively. Nonlinear constraints are called by the methods

```
opt.add_inequality_constraint(constraint,tol)
opt.add_equality_constraint(constraint,tol)
```

`tol` determines an error tolerance for the algorithm to fulfill the constraint.

5.4.2 Stopping Criteria

A stopping criterion must be chosen before the optimization can be performed. There exist different stopping criteria:

```
opt.set_stopval(stopval)
```

The algorithm stops once an objective value of at least `stopval` is found.

```
opt.set_ftol_rel(tol)
```

The algorithm stops once a relative tolerance on the change of two successive function values in the iterative process is fulfilled.

```
opt.set_ftol_abs(tol)
```

The algorithm stops once an absolute tolerance on the change of two successive function values in the iterative process is fulfilled.

```
opt.set_xtol_rel(tol)
```

The algorithm stops once a relative tolerance on the change of two successive optimization parameters in the iterative process is fulfilled.

```
opt.set_xtol_abs(tol)
```

The algorithm stops once an absolute tolerance on the change of two successive optimization parameters in the iterative process is fulfilled.

```
opt.set_maxeval(maxeval)
```

The algorithm stops once the number of function evaluations exceeds `maxeval`.

```
opt.set_maxtime(maxtime)
```

The algorithm stops once the optimization time in seconds exceeds `maxtime`.

5.4.3 Starting the Optimization

Provided all parameters have been specified in a given object `opt`, the optimization is performed by calling

```
xopt = opt.optimize(x)
```

`x` is a NumPy array of length `n` giving an initial guess for the vector of variables. The array `xopt` contains the optimized values of the optimization parameters. The optimized value of the objective function is returned by calling

```
opt_val = opt.last_optimum_value()
```

Moreover, a return code giving sucess or failure values can be retrieved by calling

```
result = opt.last_optimize_result()
```

A list of positive and negative return values can be found at [4].

6 Pasteurisation as Optimization Problem

The amount of Pasteurisation Units in a product is regulated by the product temperature, which itself is controlled by the spray water temperature. To provide a gentle but efficient thermal treatment of the product, the temperature of the spray water in each zone has to be optimized. The construction of the optimization problem is determined by structural and material reasons. The problem is solved both with the derivative-free COBYLA and the gradient-based MMA method, which are available via the NLopt library in Python.

6.1 Pasteurisation Objective Function

The optimization problem of interest in this thesis is to guarantee a special amount of PUs in the product once it leaves the tunnel pasteur. A first approach is to formulate the cost function by

$$J(u) = (\mathrm{PU}_{\mathrm{end}}(u) - \mathrm{PU}_{\mathrm{des}})^2. \tag{6.1}$$

The parameter $u \in \mathbb{R}^n$ describes the spray water temperature. In the case of the standardized tunnel pasteur introduced in Chapter 4, u is a vector with the different temperatures in either all ten zones ($n = 10$) or in the four pasteurisation zones ($n = 4$). The function $\mathrm{PU}_{\mathrm{end}}$ depends on the spray water temperature and describes the final amount of PUs in the product once it leaves the tunnel pasteur. The desired amount of PUs the product should have when the pasteurisation process is finished is determined by $\mathrm{PU}_{\mathrm{des}}$. In order to guarantee that minimizing the objective function will give its zero, the difference between $\mathrm{PU}_{\mathrm{end}}(u)$ and $\mathrm{PU}_{\mathrm{des}}$ is squared.
Figure 6.1 shows the function of the PUs which depends on the spray water temperature function. The final amount of PUs, marked with a green circle in Figure 6.1, is supposed to be optimized, i.e. should be a certain value. This is achieved by regulating the temperatures in the pasteurisation zones which are highlighted with a dotted blue ellipse in the figure.

6.2 Pasteurisation Constraints

The optimization is done with constraints; their industrial requirements and mathematical formulation is outlined in this section. It has to be emphasized that the focus of the problem formulation is on the industrial application. In consequence, some formulations

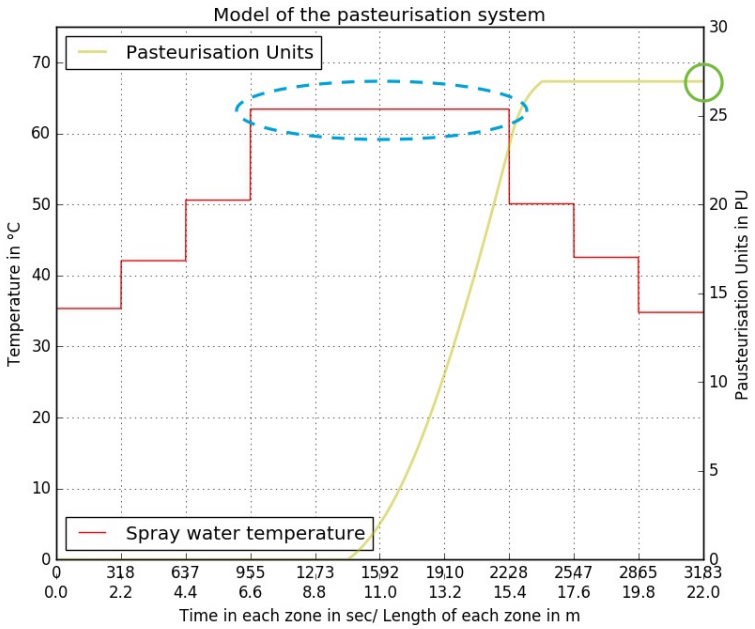


Figure 6.1: Scheme of the optimization problem

might be improvable from a mathematical and theoretical point of view, which is ignored by purpose in this section.

6.2.1 Temperature Bounds

The product is supposed to be pasteurised in a gentle way, without a change in its taste and other characteristics. This motivates to bound the temperature of the spray water from above to avoid boiling the product. A more efficient optimization algorithm is obtained if the spray water temperature is bounded from below as well. Since the pasteurisation process will not start before a minimal cut-off temperature is reached, it is reasonable to set the cut-off temperature as the lower bound. Mathematically, this constraint is expressed by

$$u \leq \text{ub}, \tag{6.2}$$

$$\text{lb} \leq u. \tag{6.3}$$

$\text{ub} \in \mathbb{R}^n$ and $\text{lb} \in \mathbb{R}^n$ describe upper and lower temperature bounds for the spray water temperature $u \in \mathbb{R}^n$. The bound constraints are given to the optimization algorithm in a separate method as described above.

6.2.2 Temperature Differences

The temperature gradient between the interior and exterior puts the container material under stress. The ability of the container to handle stress depends on the material and

restricts the temperature gap between the product and the spray water. The maximal allowed temperature difference is determined to be 25 °C. This holds for the temperature in the pasteurisation zones. In the first three heating zones, the spray water temperature is supposed to increase in order to ensure a gentle heating of the product. As the product temperature depends on the spray water temperature, it is a monotone increasing function while the product is transported through the first four zones. To measure the maximal difference between the product and the spray water temperature, it is therefore sufficient to consider the spray water temperature at the most left point of each of the pasteurisation zones. This condition is expressed by

$$u_i - u_p(t_i) \leq 25, \ i = 4:7. \tag{6.4}$$

Here, $u_p(t_i) \in \mathbb{R}$ describes the product temperature once it enters a new zone with spray water temperature $u_i \in \mathbb{R}$. The constraints need to be reformulated to be applicable in the NLopt class:

$$c_i = u_{i+3} - u_{\mathrm{p}}(t_{i+3}) - 25 \leq 0, \ i = 1:4. \tag{6.5}$$

To take care of the possible case that the spray water temperature in the pasteurisation zones is lower than the product temperature, Inequalities (6.5) are squared, yielding

$$c_i = (u_{i+3} - u_p(t_{i+3}))^2 - 25^2 \leq 0, \ i = 1:4. \tag{6.6}$$

6.2.3 Interconnected Pasteurisation Zones

Due to energy reasons, it is desirable to have equidistant temperatures in the pasteurisation zones. This is achieved by controlling the temperature difference between two successive pasteurisation zones respectively.

$$c_{i+1} = u_i - u_{i+1} = 0, \ i = 4:6. \tag{6.7}$$

Some algorithms provided in the NLopt class only support inequality constraints, thus the constraint is reformulated:

$$c_{i+1} = (u_i - u_{i+1})^2 \leq 0, \ i = 4:6. \tag{6.8}$$

6.2.4 Avoid Over-Pasteurisation

Over-pasteurisation might affect the product's taste and other characteristics. Thus, it should be avoided. To what extent over-pasteurisation has to be restricted depends on the customer. A general mathematical formulation of the constraint is described by

$$\mathrm{PU}_{\mathrm{end}}(u) \leq \omega \mathrm{PU}_{\mathrm{des}}. \tag{6.9}$$

The constant $\omega \geq 1$ indicates the amount of acceptable over-pasteurisation. The constraint needs to be reformulated for the NLopt class:

$$c_8 = \mathrm{PU}_{\mathrm{end}}(u) - \omega \mathrm{PU}_{\mathrm{des}} \leq 0. \tag{6.10}$$

6.2.5 Penalize Under-Pasteurisation

Since an under-pasteurised product is unsaleable, under-pasteurisation has to be punished. A general mathematical formulation of the constraint is given by

$$\tilde{\omega}\mathrm{PU}_{\mathrm{des}} \leq \mathrm{PU}_{\mathrm{end}}(u). \tag{6.11}$$

The constant $\tilde{\omega}$ weighs the penalty of under-pasteurisation. It should be chosen $\tilde{\omega} \geq 1$; a greater $\tilde{\omega}$ implies an increasing significance.

There exist two approaches to weigh under-pasteurisation more than over-pasteurisation. If both constraints are combined, the under-pasteurisation should be weighed more.

$$c_8 = (\mathrm{PU}_{\mathrm{end}}(u) - \omega\mathrm{PU}_{\mathrm{des}}) + c(\tilde{\omega}\mathrm{PU}_{\mathrm{des}} - \mathrm{PU}_{\mathrm{end}}(u)) \leq 0 \tag{6.12}$$

for $\omega, \tilde{\omega} \geq 1$ and a weigh constant $c > 1$.
The second approach is to include the over-pasteurisation in the objective function and retain the under-pasteurisation as a constraint.

$$\begin{aligned} &\text{minimize } J(u) = (\mathrm{PU}_{\mathrm{end}}(u) - \mathrm{PU}_{\mathrm{des}})^2 - (\omega\mathrm{PU}_{\mathrm{des}} - \mathrm{PU}_{\mathrm{end}}(u)) \\ &\text{subject to } c_8 = (\tilde{\omega}\mathrm{PU}_{\mathrm{des}} - \mathrm{PU}_{\mathrm{end}}(u))^2 \leq 0. \end{aligned} \tag{6.13}$$

6.3 Comparison of Results Given by Different Optimization Algorithms

A nonlinear optimization routine supporting both bound and nonlinear inequality constraints is required to optimize the pasteurisation process. Moreover, a local optimization algorithm is preferred. There exists one gradient-based algorithm provided in the NLopt library fulfilling these requirements, MMA. The only derivative-free algorithm supporting inequality constrains is COBYLA.
The advantage of the derivative-free COBYLA algorithm is that neither the cost nor the constraint functions have to be provided with the gradient. A numerically approximated gradient is used for the MMA algorithm since no analytical derivative can be computed.

The standardized tunnel pasteur introduced in Chapter 4 consists of ten temperature zones including four pasteurisation zones. The temperature in the recuperation zones is already optimized with respect to minimal energy consumption, therefore only the spray water temperature in the four pasteurisation zones is optimized. The following problem is solved:

$$\begin{aligned} &\text{minimize } J(u) = (\mathrm{PU}_{\mathrm{end}}(u) - \mathrm{PU}_{\mathrm{des}})^2 \\ &\text{subject to } (u_4 - u_p(t_4))^2 - 25^2 \leq 0, \\ &\qquad (u_i - u_{i+1})^2 \leq 0, \ i = 4:6, \\ &\qquad 58.0 \leq u_i \leq 65.0, \ i = 4:7. \end{aligned} \tag{6.14}$$

Due to the interconnected pasteurisation zones, it is sufficient to only regulate the temperature difference between the product and the spray water in the first pasteurisation zone. Since the optimization is supposed to compute the spray water temperature such that PU_{des} is reached, the constraints to control over- and under-pasteurisation are not included in (6.14).

The objective function is not convex, in consequence a local minimum is not necessarily a global minimum. The optimal solution of problem (6.14) differs sligthly for each algorithm. Using each result as initial guess for the other optimization algorithm will return the intial guess as optimal solution as visualized in Figure 6.2. This suggest the existence of at least two local minima.

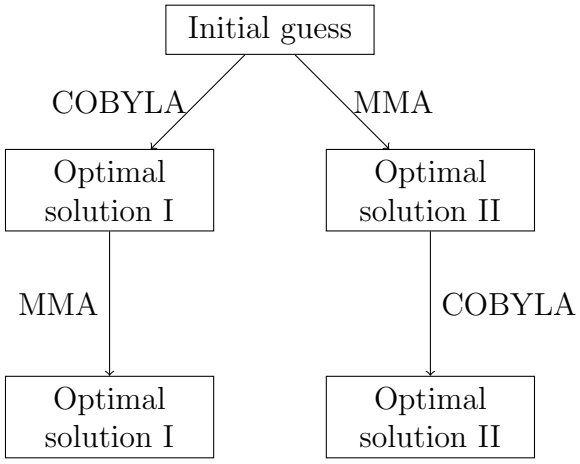


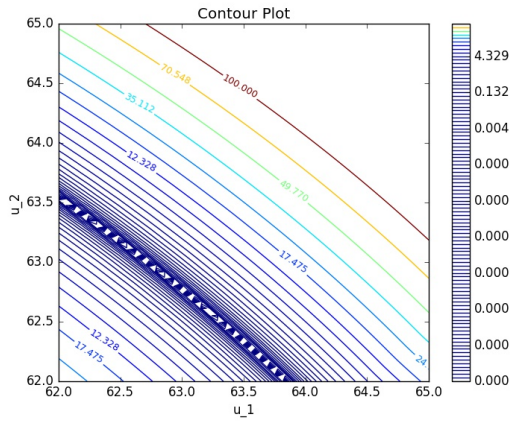
Figure 6.2: Comparison of optimal solutions using different algorithms and initial guesses

A two dimensional contour plot is created in order to visualize the level curves of the cost function, assuming the temperatures in the first two and the second two pasteurisation zones are coupled, respectively. Figure 6.3 proves the existence of several minima for the cost function defined in problem (6.14): the level set at 0 is a line.
Figure 6.4 shows the results for problem (6.14) with initial guess [63.0, 63.0] for both algorithms. The red line marks the set of feasible points. The gradient-based routine fails to find an optimal solution, while the derivative-free algorithm succeeded. Reasons for the failure of the MMA algorithm to solve problem (6.14) are discussed in the next section.

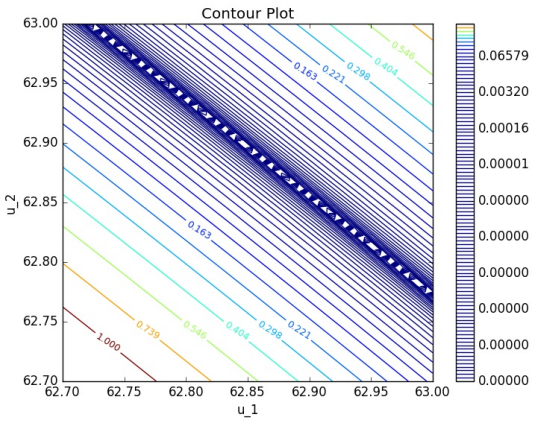
6.4 MMA and Equality Constraints

The method of moving asymptotes fails to solve problem (6.14). This is caused by the construction of the algorithm and the problem formulation. The constraint to couple the temperatures in the pasteurisation zones is formulated as

$$(u_i - u_{i+1})^2 \leq 0, \; i = 4:6. \tag{6.15}$$

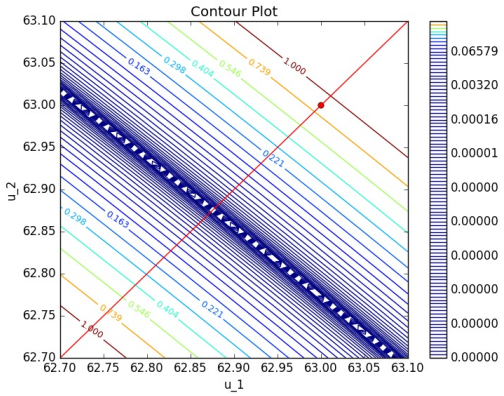


(a) Contour plot on the interval [62.0, 65.0]

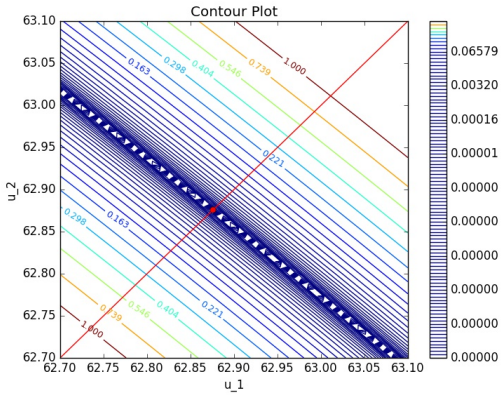


(b) Zoomed-in contour plot

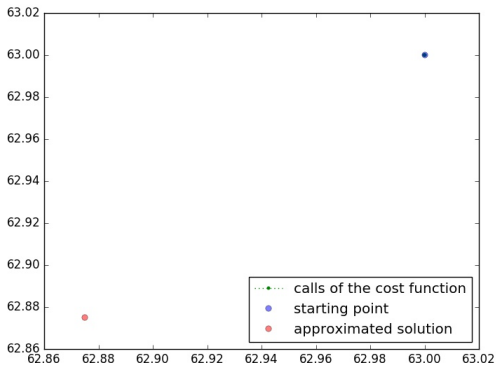
Figure 6.3: Contour plot of the objective function given in (6.14)



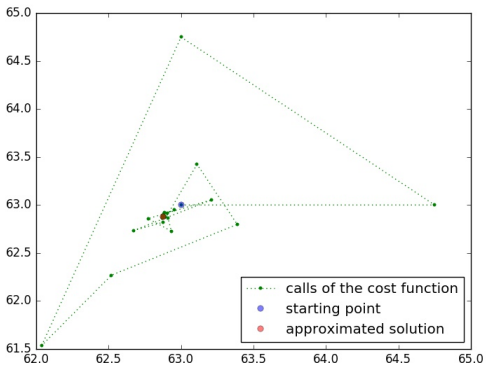
(a) Optimal solution found with MMA



(b) Optimal solution found with COBYLA



(c) Iteration results for MMA



(d) Iteration results for COBYLA

Figure 6.4: Results of the optimization algorithms for initial guess [63.0, 63.0]

This indirectly requires the MMA routine to generate and solve convex subproblems for equality constraints.
Assume the MMA method also supports equality constraints. The approximating functions for the cost and the constraint function at iteration step (k,ℓ) are defined in Subsection 5.2.3 by

$$g_i^{(k,\ell)}(x) = \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k,\ell)}}{u_j^{(k)} - x_j} + \frac{q_{ij}^{(k,\ell)}}{x_j - l_j^{(k)}} \right) + r_i^{(k,\ell)}$$

with $p_{ij}^{(k,\ell)}$, $q_{ij}^{(k,\ell)}$ and $r_i^{(k,\ell)}$ independend of x. Let the constraints with indices $j = 1 : m_e$ be equality constraints and the constraints with indices $j = m_e + 1 : m$ be inequality constraints. The Lagragian function for the subproblem at iteration (k,ℓ) is given by

$$L^{(k,\ell)}(x;\lambda) = g_0^{(k,\ell)}(x) + \sum_{j=1}^{m_e} \lambda_j^{(k,\ell)} g_j^{(k,\ell)}(x) + \sum_{j=m_e+1}^{m} \lambda_j^{(k,\ell)} g_j^{(k,\ell)}(x). \tag{6.16}$$

In order to verify if the generated subproblem is convex, the Hessian matrix of the Lagrangian function is evaluated.

$$\begin{aligned} \frac{\partial^2 L^{(k,\ell)}}{\partial x_i^2}(x;\lambda) = & \frac{2}{(u_i^{(k)} - x_i)^3} \Big(p_{0i}^{(k,\ell)} + \sum_{j=1}^{m_e} \lambda_j^{(k,\ell)} p_{ji}^{(k,\ell)} + \sum_{j=m_e+1}^{m} \lambda_j^{(k,\ell)} p_{ji}^{(k,\ell)} \Big) + \\ & \frac{2}{(x_i - l_i^{(k,\ell)})^3} \Big(q_{0i}^{(k,\ell)} + \sum_{j=1}^{m_e} \lambda_j^{(k,\ell)} q_{ji}^{(k,\ell)} + \sum_{j=m_e+1}^{m} \lambda_j^{(k,\ell)} q_{ji}^{(k,\ell)} \Big). \end{aligned} \tag{6.17}$$

There exist no regulations on the equality terms to keep the numerator positive since $\lambda_j^{(k,\ell)}$ can be negative for $j = 1 : m_e$. In consequence, the approximation is not necessarily convex.

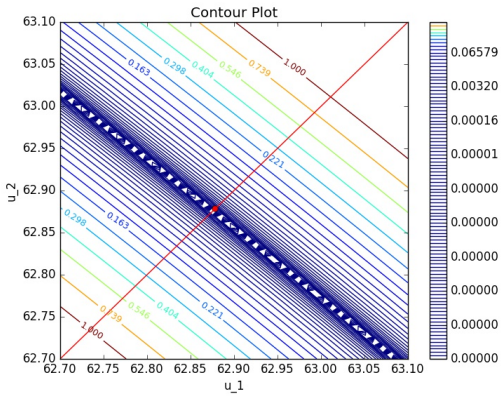
This problem can be solved in different ways. Either by changing the problem description or by modifying the method of moving asymptotes such that it can be provided with equality constraints. Both methods are presented in the following sections.

6.4.1 Modification of the Problem Description

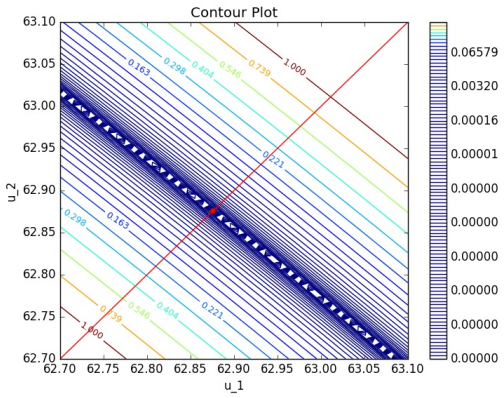
The formulation of the constraint in (6.15) gives a very narrow feasible region, in fact the feasible region is a straight line. The constraint needs to be reformulated in order to broaden the feasible region. The optimization problem becomes

$$\begin{aligned} &\text{minimize } J(u) = (\mathrm{PU}_{\mathrm{end}}(u) - \mathrm{PU}_{\mathrm{des}})^2 \\ &\text{subject to } (u_i - u_p(t_i))^2 - 25^2 \le 0, \ i = 1:4, \\ &\qquad u_i - u_{i+1} - 10^{-7} \le 0, \ i = 4:6, \\ &\qquad u_{i+1} - u_i - 10^{-7} \le 0, \ i = 4:6, \\ &\qquad 58.0 \le u_i \le 65.0, \ i = 4:7. \end{aligned} \tag{6.18}$$

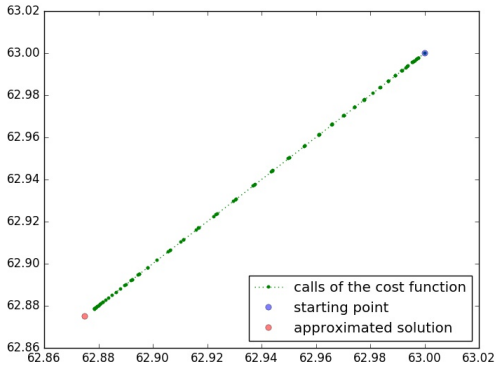
Figure 6.5 shows the approximated optimal solution for each iteration step both for the MMA routine and for the COBYLA algorithm for initial guess [63.0, 63.0].



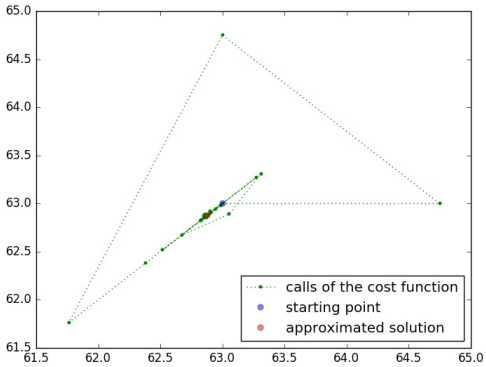
(a) Optimal solution found with MMA



(b) Optimal solution found with COBYLA



(c) Iteration results for MMA

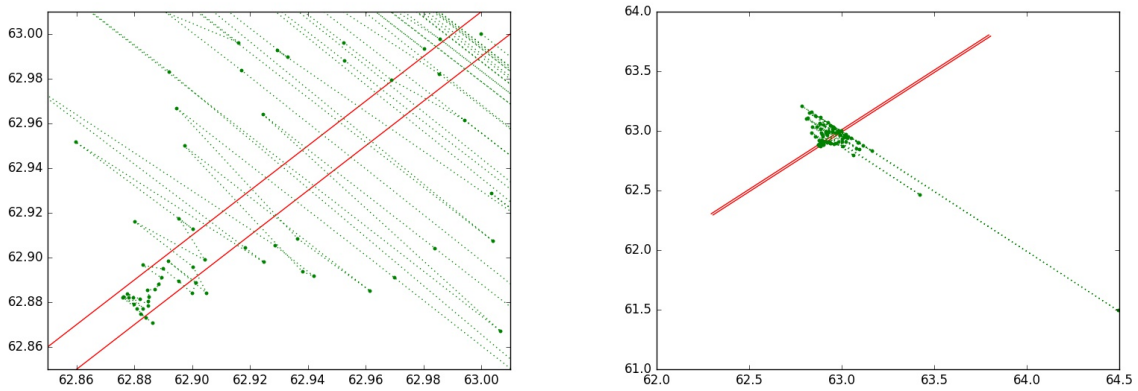


(d) Iteration results for COBYLA

Figure 6.5: Results of the optimization algorithms for initial guess [63.0, 63.0]

Since the feasible region is not a line, problem (6.18) might have several optimal solutions. If the difference is very small, in this case smaller or equal than $2 \cdot 10^{-7}$, this is negligible in the application of beer pasteurisation.
Using the example to optimize the pasteurisation process, the approximating scheme of the MMA algorithm is made visible. The feasible region and the optimal solutions of the subproblem printed at every call of the cost function are stretched by the factor $5 \cdot 10^5$, see Figure 6.6. Caused by the still very narrow feasible region, the MMA algorithm violates the feasibility condition during the search for an optimal solution.

To get a unique optimal solution, the cost function has to be modified. The constraint



(a) Iteration results in the interval [62.3, 63.8] **(b)** Iteration results

Figure 6.6: Iteration results for MMA zoomed with factor $5 \cdot 10^5$

(6.15) can be put into the cost function and the optimization problem becomes

$$\begin{aligned} &\text{minimize } J(u) = (\mathrm{PU}_{\mathrm{end}}(u) - \mathrm{PU}_{\mathrm{des}})^2 + (u_5 - u_6)^2 \\ &\text{subject to } (u_i - u_p(t_i))^2 - 25^2 \le 0, \ i = 1:4, \\ &\qquad 58.0 \le u_i \le 65.0, \ i = 4:7. \end{aligned} \tag{6.19}$$

This formulation has the advantage that the rebuilt cost function will have a unique minimum as it can be seen in the contour plot, Figure 6.7. The second modification is preferred since a unique optimal solution of the problem can be found. In the first modification of the original problem there level set at zero is a line. Combined with a broadened feasible region, the algorithms might find different solutions depending on the initial guess and the width of the feasible region.

6.4.2 Modification of the Algorithm

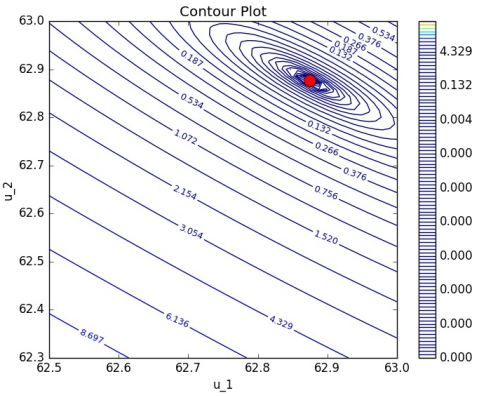
Bletzinger suggests in [8] a method to implement the approximations for equality constraints in order to obtain a convex subproblem. Instead of using the approximations (5.18) in Subsection 5.2.3, equality constraints should be approximated by functions of the form

$$g_i^{(k,\ell)}(x) = \sum_{j=1}^{n} \left(\frac{p_{ij}^{(k,\ell)}}{u_j^{(k)} - x_j} - \frac{p_{ij}^{(k,\ell)}}{x_j - l_j^{(k)}} \right) + r_i^{(k,\ell)} \tag{6.20}$$

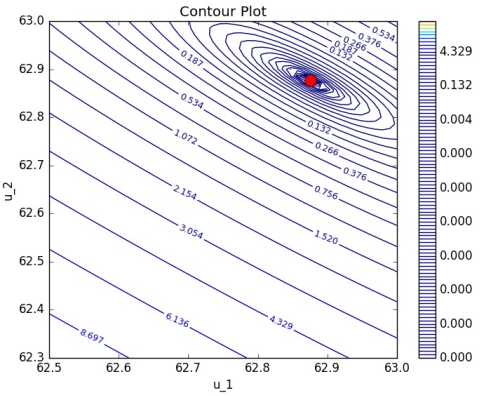
for

$$p_{ij}^{(k,\ell)} = \frac{1}{2}(u_j^{(k)} - x_j)^2 \frac{\partial c_i}{\partial x_j} = \frac{1}{2}(x_j - l_j^{(k)})^2 \frac{\partial c_i}{\partial x_j}, \tag{6.21}$$

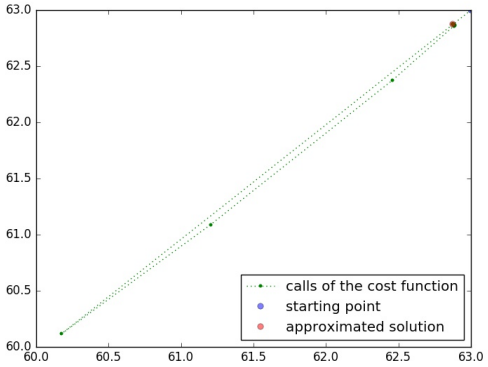
$$r_i^{(k,\ell)} = c_i(x^{(k)}). \tag{6.22}$$



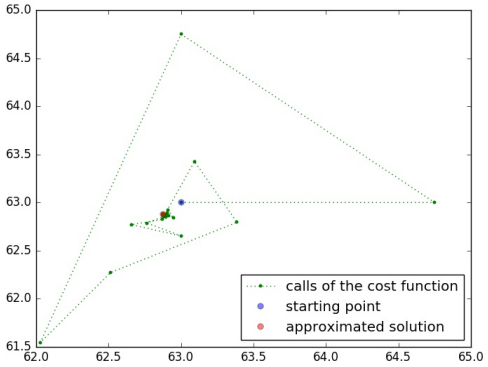
(a) Optimal solution found with MMA



(b) Optimal solution found with COBYLA



(c) Iteration results for MMA



(d) Iteration results for COBYLA

Figure 6.7: Optimal solution for initial guess [63.0, 63.0] given by the red point

The Lagrangian function of the subproblem at iteration (k, ℓ) is given by

$$L^{(k,\ell)}(x;\lambda) = g_0^{(k,\ell)}(x) + \sum_{j=1}^{m_e} \lambda_j^{(k,\ell)} g_j^{(k,\ell)}(x) + \sum_{j=m_e+1}^{m} \lambda_j^{(k,\ell)} g_j^{(k,\ell)}(x). \tag{6.23}$$

A necessary condition to find an optimal solution is

$$\frac{\partial L^{(k,\ell)}}{\partial x_i}(x;\lambda) = \frac{P_i + P_{ei}}{(u_i^{(k)} - x_i)^2} - \frac{Q_i - P_{ei}}{(x_i - l_i^{(k)})^2} = 0, \tag{6.24}$$

with the abbreviations

$$P_i = p_{0i}^{(k)} + \sum_{j=m_e+1}^{m} \lambda_j^{(k,\ell)} p_{ji}^{(k,\ell)}(x), \tag{6.25}$$

$$P_{ei} = \sum_{j=1}^{m_e} \lambda_j^{(k,\ell)} p_{ji}^{(k,\ell)}(x), \tag{6.26}$$

$$Q_i = q_{0i}^{(k)} + \sum_{j=m_e+1}^{m} \lambda_j^{(k,\ell)} q_{ji}^{(k,\ell)}(x). \tag{6.27}$$

This requires that both numerators are either positive or negative,

$$\left.\begin{aligned} P_i + P_{ei} > 0 \\ Q_i - P_{ei} > 0 \end{aligned}\right\} \quad \Rightarrow -P_i < P_{ei} < Q_i \tag{6.28}$$

$$\left.\begin{aligned} P_i + P_{ei} < 0 \\ Q_i - P_{ei} < 0 \end{aligned}\right\} \quad \Rightarrow Q_i < P_{ei} < -P_i. \tag{6.29}$$

By construction, both P_i and Q_i are always positive, in consequence condition (6.29) is impossible. In order to have a necessary condition for a minimum, condition (6.28) must hold. The Hessian of the Lagrange function for the subproblem is given by

$$\frac{\partial^2 L^{(k,\ell)}}{\partial x_i^2}(x;\lambda) = \frac{2(P_i + P_{ei})}{(u_i^{(k)} - x_i)^3} + \frac{2(Q_i - P_{ei})}{(x_i - l_i^{(k)})^3} > 0. \tag{6.30}$$

This shows that the modified approximations (6.20) generate convex subproblems.
In the case of mixed signs of numerators, the Lagrangian function is monotonously inceasing or decreasing for $l_j^{(k)} < x_j < u_j^{(k)}$.

6.5 Results

Table 6.1 shows the number of cost function calls by the MMA and the COBYLA routine, and the scale of the optimal solution obtained with both algorithms. Only temperatures in the pasteurisation zones are optimized under the assumption that they are coupled pairwise. Starting with initial guess $[63.0, 63.0]$ and tolerance 10^{-8} for the algorithms to fulfill the constraints, the stopping criterion is a relative tolerance on the input given by 10^{-5}. The results of the table are visualized in Figure 6.8.

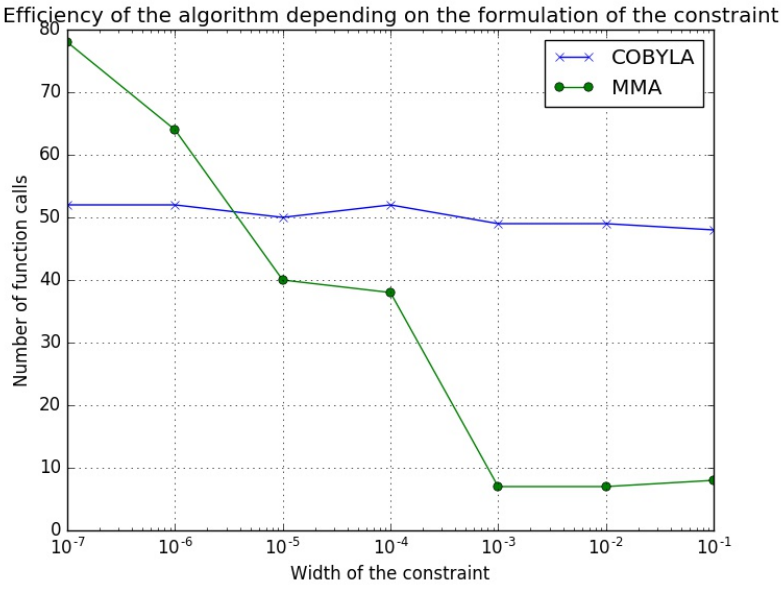


Figure 6.8: Scheme of the optimization problem

	COBYLA		MMA	
Type of the constraint handling	Calls of the cost function	Level of the cost function	Calls of the cost function	Level of the cost function
as equality constraint	48	10^{-9}	5	10^{-1}
by penalty terms	89	10^{-9}	21	10^{-6}
feasible region 10^{-1}	48	10^{-9}	8	10^{-8}
feasible region 10^{-2}	49	10^{-9}	7	10^{-9}
feasible region 10^{-3}	49	10^{-9}	7	10^{-7}
feasible region 10^{-4}	52	10^{-8}	38	10^{-16}
feasible region 10^{-5}	50	10^{-8}	40	10^{-6}
feasible region 10^{-6}	52	10^{-8}	64	10^{-4}
feasible region 10^{-7}	52	10^{-8}	78	10^{-4}

Table 6.1: Comparison of two different optimization algorithms. Note the influence of the width of the feasible region for the performance of MMA

The MMA algorithm fails to find an optimal solution with equality constraint; in the case of a very narrow feasible region, the cost function is called more frequently without improving the accuracy of the solution. The correlation between the width of the feasible region and the accuracy of the optimal solution is shown in Table 6.1. The smaller the feasible region is chosen, the more calls of the cost function can be noticed, visualized in Figure 6.8. These calls are barely noticeable improving the approximated result and thus make the method ineffective in computational and time effort. Having a wide feasible region, i.e. 10^{-1}, gives a quite accurate result after a reasonable number of calls.

The COBYLA algorithm gives stable results with accuracy 10^{-8} or 10^{-9}, independent of the width of the feasible region and even for equality constraints. The number of cost function calls by the COBYLA method shown in Table 6.1 indicates that the algorithm calls the cost function approximately 50 times independent of the width if the feasible region. The number of calls increases noticeably if the cost function is modified. In this case, the derivative-free method is ineffective in its search and beaten by the gradient-based routine.
In order to have competitive algorithms, the MMA algorithm needs to be implemented with an extention for supporting equality constraints. The MMA routine is preferrable since it is more efficient in the search for an optimal solution in case the problem is formulated correctly. The disadvantage of the gradient-based method is the challenge of narrow feasible regions. Constraints of this type will make the algorithm fail to find an optimal solution. The COBYLA method will generate stable solutions with a comparable number of iterations, independent of the feasible region.

In this particular context, the gradient-based MMA algorithm is a serious alternative to derivative-free methods. It depends on the goal of the optimization which method is preferrable. The computational effort increases when the gradient has to be computed numerically but improves the effectiveness of the search method.

7 Cold Spot Simulation

The model for the theraml processes used so far assumes a perfectly mixed product. In reality, heat convection occurs within the product. The existence of a cold spot in the liquid product caused by the fluid flow due to convection is described in Chapter 3. The cold spot is the last region to react on heat energy. In this chapter, a numerical solution is developed to visualize the existence of a cold spot. This chapter serves as a preparation for future work; the heat model presented here needs to be simplified in order to reduce the computational effort. An optimization based on this model is subject of further research.

7.1 Heat Convection

Let $u(x,t)$ describe the temperature in the object at position x and time t. The heat equation in n dimensions is given by the Fourier equation

$$\rho c \frac{\partial u(x,t)}{\partial t} - \nabla \cdot (k \nabla u(x,t)) = g(x), \;\; x \in [0,L]^n \subseteq \mathbb{R}^n \tag{7.1}$$

with density ρ in kg/m^3, heat capacity c in J/(kg K) and thermal coefficient k in W/(m K); all values depend on temperature and pressure. The function g denotes an external heat source. The initial condition

$$u(x,0) = f(x), \;\; \forall x \in [0,L]^n \subseteq \mathbb{R}^n$$

describes the temperature of the object before the heating process at every point of the domain. Boundary conditions on the boundary points of the domain specify the heat exchange between the outside and the inside of the domain.
The calculations can be simplified assuming that heat capacity and thermal coefficient are independent of the temperature and no external heat source exists. The heat equation becomes

$$\frac{\partial u(x,t)}{\partial t} - D \Delta u(x,t) = 0 \tag{7.2}$$

with the thermal diffusitivity $D = \frac{k}{\rho c}$, [7, p.217 ff.].
An analytical solution of this partial differential equation is derived in the next section without giving very detailed explanations. The focus lies on numerical solutions of Equation (7.2), which are used to model the thermal processes of the product in a container.

7.1.1 Analytical Solution

Equation (7.2) can be solved analytically by separation of variables. Writing the temperature function $u(x,t)$ as the product of a function $X(x)$ dependent on space and a function $T(t)$ dependent on time, gives

$$u(x,t) = X(x) \cdot T(t). \tag{7.3}$$

The simple heat equation (7.2) can be expressed by

$$\frac{\mathrm{d}T(t)}{\mathrm{d}t} X(x) = D \frac{\mathrm{d}^2 X(x)}{\mathrm{d}x^2} T(t), \tag{7.4}$$

and reformulation yields

$$\frac{\mathrm{d}T(t)}{T(t)\mathrm{d}t} = D \frac{\mathrm{d}^2 X(x)}{X(x)\mathrm{d}x^2}. \tag{7.5}$$

Since the left hand side only depends on time t and the right hand side only depends on space x, it can be concluded that both sides must equal a constant $-b^2$:

$$\begin{aligned} \frac{\mathrm{d}T(t)}{\mathrm{d}t} &= -b^2 T(t), \\ D \frac{\mathrm{d}^2 X(x)}{\mathrm{d}x^2} &= -b^2 X(x), \end{aligned}$$

giving the solutions

$$\begin{aligned} T(t) &= A e^{-b^2 t}, \\ X(x) &= B \cos(bx) + C \sin(bx). \end{aligned}$$

The general analytical solution of Equation (7.2) is given by

$$u(x,t) = (D \cos(bx) + E \sin(bx)) e^{-b^2 t}. \tag{7.6}$$

A particular solution is found by taking into account the initial and boundary conditions, for further discussion the reader is referred to [7].

7.1.2 Numerical Solution

To simulate the thermal processes, the heat equation is solved by numerical methods. Due to the numerical computations, errors can accumulate and instability can occur. Therefore, numerical solution methods have to be chosen carefully.
There exists different numerical methods to approximate the solution of Equation (7.2). Each method has advantages and disadvantages; it depends on the problem description, which method is the best to be chosen.

Crank-Nicolson Method

The solutions of the partial derivatives

$$\frac{\partial f(x,t)}{\partial x} \text{ and } \frac{\partial^2 f(x,t)}{\partial x^2}$$

can be approximated numerically by the finite difference method, FDM. This method replaces the derivative by approximated values. These values are obtained by evaluating the Taylor series at certain discrete points, called nodal points. [7]

Given a function $f(x,t)$ defined for $x \in [L,R]$ and $t \geq 0$, the interval $[L,R]$ is discretized with N internal grid points, using the notation $x_i = L + i\Delta x$ for $i = 0,\ldots,N+1$ for $\Delta x = \frac{R-L}{N+1}$. Taylor series expansion of f around x gives

$$f(x+\Delta x,t) = f(x,t) + \Delta x \frac{\partial f(x,t)}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 f(x,t)}{\partial x^2} + \mathcal{O}(\Delta x^3) \tag{7.7}$$

and

$$f(x-\Delta x,t) = f(x,t) - \Delta x \frac{\partial f(x,t)}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 f(x,t)}{\partial x^2} + \mathcal{O}(\Delta x^3) \tag{7.8}$$

For small Δx, all terms Δx^n for $n \geq 3$ can be neglected, and subtracting Equation (7.8) from Equation (7.7) and reformulating the result leads to

$$\frac{\partial f(x,t)}{\partial x} \approx \frac{f(x+\Delta x,t) - f(x-\Delta x,t)}{2\Delta x}. \tag{7.9}$$

Equation (7.9) approximates the first derivative of the function f with an error term of $\mathcal{O}(\Delta x^3)$, it is called an approximation of second order accuracy.

The finite difference approximation for the second derivative is deduced by adding Equation (7.8) to Equation (7.7) and reformulating the result:

$$\frac{\partial^2 f(x,t)}{\partial x^2} \approx \frac{f(x+\Delta x,t) - 2f(x,t) + f(x-\Delta x,t)}{\Delta x^2}. \tag{7.10}$$

Equation (7.10) is again of second order.

The finite difference approximations of the first and second derivatives can be written more compactly in matrix form:

$$\frac{\partial f}{\partial x} \approx \frac{1}{2\Delta x}\begin{pmatrix} -1 & 0 & 1 & & & \\ 0 & -1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 & 0 \\ & & & -1 & 0 & 1 \end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \\ f_{N+1} \end{pmatrix} \in \mathbb{R}^{N\times N+2}$$

and

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{1}{\Delta x^2} \begin{pmatrix} 1 & -2 & 1 & & & \\ 0 & 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & 0 \\ & & & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \\ f_{N+1} \end{pmatrix} \in \mathbb{R}^{N \times N+2}.$$

The matrices have to be modified depending on different boundary conditions, and are always of sparse, tridiagonal form. There exist common boundary conditions for parabolic partial differential equations [11]:

- **Dirichlet boundary condition**: f is specified at $x = a$,

$$f(a,t) = \alpha(t) \tag{7.11}$$

- **Neumann boundary condition**: $\frac{\partial f}{\partial x}$ is specified at $x = a$,

$$\frac{\partial f}{\partial x}(a,t) = \gamma(t) \tag{7.12}$$

- **Robin boundary condition**: Linear combination of f and $\frac{\partial f}{\partial x}$ is specified at $x = a$,

$$\frac{\partial f}{\partial x}(a,t) = \mu(f(a,t) - f_{\text{out}}(t)) \tag{7.13}$$

  with μ a constant and $f_{\text{out}}(t)$ a given function.

In the case of the heat equation (7.1), the boundary conditions have a physical meaning in terms of temperature:

- The Dirichlet condition implies that the temperature $\alpha(t)$ is known at the boundary point $x = a$.
- The Neumann condition implies that the heat flux is known at the boundary point $x = a$. Moreover, $\frac{\partial f}{\partial x}(a,t) = 0$ tells that the point $x = a$ is isolated from the environment.
- The Robin condition implies that the heat flux at $x = a$ is proportional to the temperature difference $f(a,t) - f_{\text{out}}(t)$.

The partial derivative with respect to time

$$\frac{\partial f(x,t)}{\partial t},\ t \geq 0$$

can be approximated by the implicit trapezoidal rule. One time step is assumed to have length Δt. Using the notation $f_i^n = f(i\Delta x, n\Delta t)$, it is a combination of the explicit forward Euler method

$$f_i^{n+1} = f_i^n + \Delta t \frac{\partial f_i^n}{\partial t}$$

and the explicit backward Euler method

$$f_i^{n+1} = f_i^n + \Delta t \frac{\partial f_i^{n+1}}{\partial t}.$$

The trapezoidal rule is formulated by

$$f_i^{n+1} = f_i^n + \frac{\Delta t}{2}\left(\frac{\partial f_i^n}{\partial t} + \frac{\partial f_i^{n+1}}{\partial t}\right).$$

Given a partial differential equation of the form

$$\frac{\partial f(x,t)}{\partial t} = \frac{\partial^2 f(x,t)}{\partial x^2},$$

the finite difference method for the space discretization and the trapezoidal rule for the time discretization are combined. This leads to the Crank-Nicolson method:

$$f_i^{n+1} = f_i^n + \frac{\Delta t}{2}\left(\frac{f_{i+1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}}{\Delta x^2} + \frac{f_{i+1}^n - 2f_i^n + f_{i-1}^n}{\Delta x^2}\right) \tag{7.14}$$

With the matrix form for the space discretization, the Crank-Nicolson method becomes

$$\mathrm{I} f^{n+1} = \mathrm{I} f^n + \frac{\Delta t}{2\Delta x^2}(T f^{n+1} + T f^n). \tag{7.15}$$

The sparse linear system

$$(\mathrm{I} - \frac{\Delta t}{2\Delta x^2} T) f^{n+1} = (\mathrm{I} + \frac{\Delta t}{2\Delta x^2} T) f^n \tag{7.16}$$

has to be solved for every time step.

7.2 Examples

In this section, the heat equation is solved in the one- and two-dimensional case, both time-dependent and time-independent. This is a first attemp to model numerically the thermal processes in a container filled with a liquid product during the thermal treatment in the tunnel pasteur. The initial and boundary conditions are choosen accordingly.

7.2.1 One-Dimensional Case

In the one-dimensional case, the heat equation without external heat source is given by

$$\frac{\partial u(x,t)}{\partial t} - D\Delta u(x,t) = 0, \;\; x \in [0, L] \subseteq \mathbb{R}. \tag{7.17}$$

This is a very simplified one-dimensional model of a container with beer of length L and width 0, visualized in Figure 7.1.
The initial condition describes the temperature of the product before the pasteurisation process. The Dirichlet boundary condition at $x = L$ describes the outer heating temperature from above, in this model the temperature of the spray water. The boundary condition at $x = 0$ is chosen to be a Neumann condition to describe the heat flux between container and conveyor belt.

Figure 7.1: Bar of length L

Steady Heat Equation

In the case of the steady heat equation, the temperature function u is independent of time t, more precisely

$$\frac{\partial u}{\partial t} = 0.$$

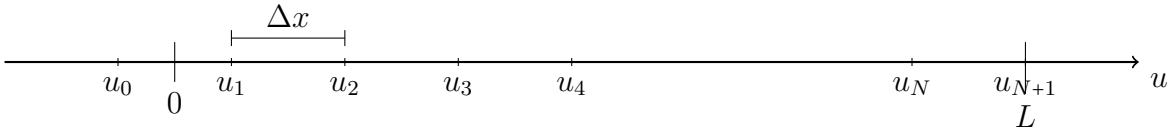
Thus, it is preferrable to write $u(x,t) = u(x)$ for all $t \geq 0$. The steady heat equation is given by the Laplace equation

$$D\Delta u = 0 \tag{7.18}$$

with mixed boundary conditions

$$\frac{\partial u}{\partial x}(0) = \alpha,\ u(L) = \beta. \tag{7.19}$$

In the one-dimensional case, $\Delta u = \frac{\partial^2 u}{\partial x^2}(x)$. Having N internal grid points of distance $\Delta x = \frac{L}{N+0.5}$ in the interval $[0, L]$ as sketched below,



and using the notation $u_i = u(i\Delta x)$ for $i = 1 : N+1$, the second order finite difference approximation is given by

$$\Delta u = \frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2},\ i = 1 : N. \tag{7.20}$$

With Taylor series expansion at $x = 0$, the Neumann boundary condition can be approximated of second order accuracy with

$$\frac{\partial u}{\partial x}(0) = \frac{u_0 - u_1}{\Delta x} = \alpha.$$

In the case of a container in a tunnel pasteur, it is assumed no heat flux between the container and the conveyor belt takes place, thus $\alpha = 0$. This gives $u_0 = u_1$. With the notation

$$T_{\Delta x} = \frac{1}{\Delta x^2}\begin{pmatrix} -1 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{N\times N},\ b_{\Delta x} = \frac{1}{\Delta x^2}\begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{pmatrix} \in \mathbb{R}^N,$$

the numerical solution of Laplace equation is given by the solution of the sparse linear system

$$DT_{\Delta x}u = -Db_{\Delta x}. \tag{7.21}$$

The heat plot of the solution can be seen in Figure 7.2 for spray water temperature of β = 60 °C and heat flux of α = 0 °C. For step size of length $\Delta x = \frac{1}{40.5}$ the numerical

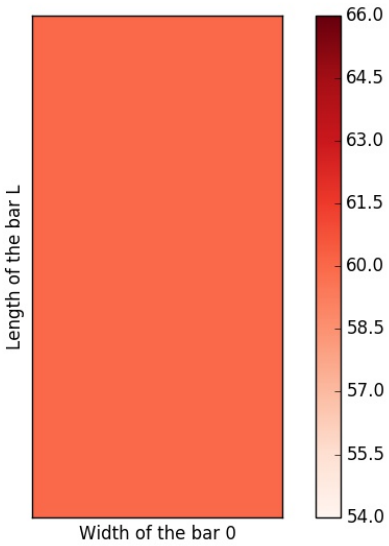


Figure 7.2: Heat map for the simple model of a container in the stationary case, heated at 60 °C

solution approximates the analytic solution quite well, as shown in Figure 7.3.

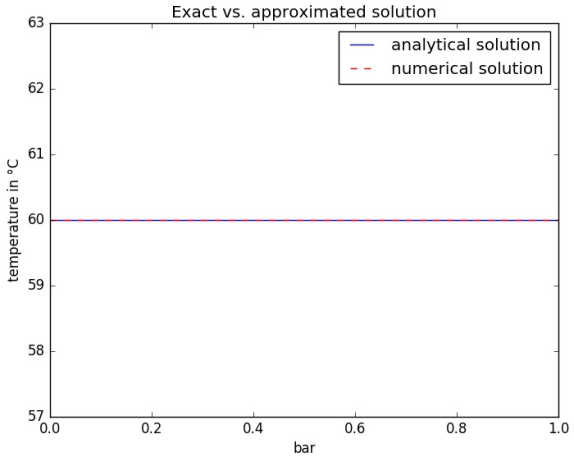


Figure 7.3: Numerical and analytical solution for space step length 0.02

Dynamic Heat Equation

In the dynamic case, the heat equation is dependent on time, more precisely

$$\frac{\partial u(x,t)}{\partial t} - D\Delta u(x,t) = 0,\ x \in [0,L],\ t \geq 0. \tag{7.22}$$

In addition to the boundary conditions

$$\frac{\partial u}{\partial x}(0,t) = \alpha,\ u(L,t) = \beta,\ t \geq 0, \tag{7.23}$$

there exists an initial condition

$$u(x,0) = f(x),\ x \in [0,L], \tag{7.24}$$

to describe the temperature of the product at the begin of the computation. The Laplacian part of the one-dimensional non-steady heat equation is approximated as in the steady case. Equation (7.22) becomes

$$\frac{\partial u}{\partial t} = D(T_{\Delta x}u + b_{\Delta x}) \tag{7.25}$$

with, assuming again $\alpha = 0$,

$$T_{\Delta x} = \frac{1}{\Delta x^2}\begin{pmatrix} -1 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{N\times N},\ b_{\Delta x} = \frac{1}{\Delta x^2}\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \beta \end{pmatrix} \in \mathbb{R}^N.$$

The A-stable trapezoidal rule is applied for the time discretization.

$$u^{n+1} = u^n + \frac{\Delta t}{2}(u^n + u^{n+1}) \ \Leftrightarrow\ (1 - \frac{\Delta t}{2})u^{n+1} = (1 + \frac{\Delta t}{2})u^n.$$

Combining this equation with the space discretization of the Laplacian gives the Crank-Nicolson method

$$(\mathrm{I} - \frac{\mu}{2}T)u^{n+1} = (\mathrm{I} + \frac{\mu}{2}T)u^n + \mu b, \tag{7.26}$$

with $\mu = D\frac{\Delta t}{\Delta x^2}$.

Figure 7.4 shows the heat map for a bar of length 1, $\Delta x = \frac{1}{40.5}$, $\Delta t = 0.01$ s, $D = 1$, Neumann condition $\alpha = 0$ °C, Dirichlet condition $\beta = 60$ °C and an initial temperature of 18 °C. As the can has width length 0, the spray water temperature affects the product temperature just from above.

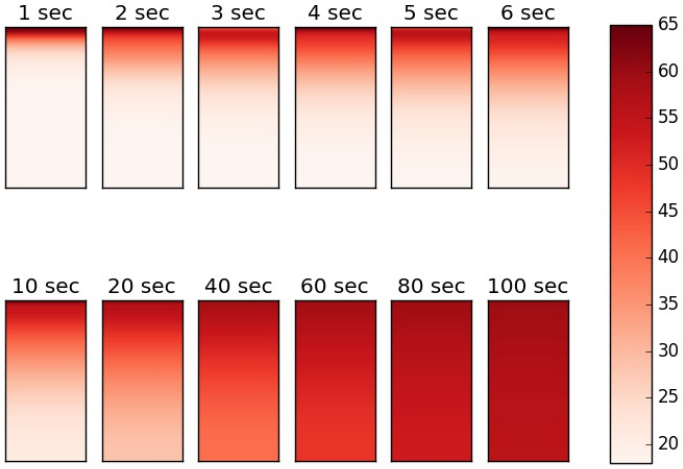


Figure 7.4: Heat distribution at different times

7.2.2 Two-Dimensional Case

The two-dimensional case allows a more accurate description of a container. A two-dimensional model describes one slice of the container as skeched in Figure 7.5.

Figure 7.5: Plate of width K and length L

The two-dimensional heat equation is given by

$$\frac{\partial u(x,y,t)}{\partial t} - D\Delta u(x,y,t) = 0,\ x \in [0,K],\ y \in [0,L],\ t \geq 0 \tag{7.27}$$

with D a positive constant for the thermal diffusivity. The function $u(x,y,t)$ describes the temperature of the plate at position (x,y) and time t. The Dirichlet boundary conditions

$$u(0,y,t) = \alpha,\ u(K,y,t) = \beta,\ y \in [0,L],\ t \geq 0, \tag{7.28}$$
$$u(x,0,t) = \gamma,\ u(x,L,t) = \delta,\ x \in [0,K],\ t \geq 0 \tag{7.29}$$

specify the temperature along the outer walls of the container. In the pasteurisation model, δ is the spray water temperature, α and β describe the temperature along the

sides, for simplicity also the spray water temperature, and γ describes the temperature flux between the container and the conveyor belt. The temperature of the product before the thermal treatment is given by the initial condition

$$u(x, y, 0) = f(x, y),\ (x, y) \in [0, L] \times [0, K].$$

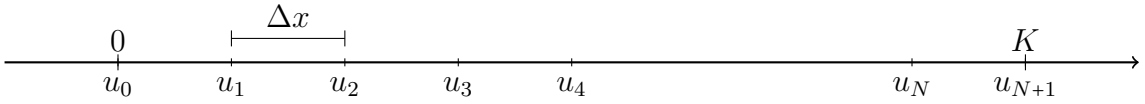
The analytic solution can be derived as in the one-dimensional case by separation of variables and taking care of the initial and boundary conditions. Numerical approximations of the solutions are derived in the following sections.

Steady Heat Equation

In the steady case, the heat equation in two dimensions becomes

$$D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0. \tag{7.30}$$

The space needs to be discretized both in x- and y-direction for the finite difference method.
In x-direction, Dirichlet boundary conditions are given at 0 and K. With N internal grid points and step size $\Delta x = \frac{K}{N+1}$, the interval $[0, K]$ is discretized as sketched below.



The finite difference approximation

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2},\ i = 1, \ldots, N$$

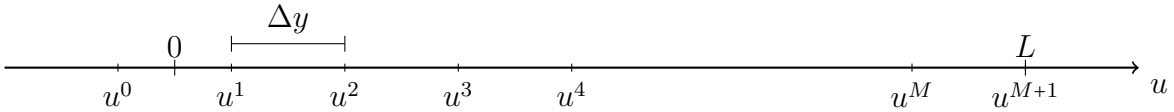
can be written more compactly

$$\frac{\partial^2 u}{\partial x^2} = T_{\Delta x} u + b_{\Delta x}$$

with

$$T_{\Delta x} = \frac{1}{\Delta x^2} \begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{N \times N},\ b_{\Delta x} = \frac{1}{\Delta x^2} \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{pmatrix} \in \mathbb{R}^N.$$

In y-direction, Neumann boundary condition at 0 and Dirichlet boundary condition at L are given. With M internal grid points, the interval $[0, L]$ is discretized with step length $\Delta y = \frac{L}{M+0.5}$ as skeched below.



The finite difference approximation

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u^{j-1} - 2u^j + u^{j+1}}{\Delta x^2},\ j = 1, \ldots, M$$

can be written more compactly, assuming $\gamma = 0$,

$$\frac{\partial^2 u}{\partial x^2} = S_{\Delta y} u + d_{\Delta y}$$

with

$$S_{\Delta y} = \frac{1}{\Delta y^2} \begin{pmatrix} -1 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{M \times M},\ d_{\Delta y} = \frac{1}{\Delta y^2} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \delta \end{pmatrix} \in \mathbb{R}^M.$$

Combining the discretizations in x- and y-direction, the heat equation can be approximated numerically by solving the sparse linear system

$$Au = c \tag{7.31}$$

with

$$A = \begin{pmatrix} B & \frac{\mathrm{I}}{\Delta y^2} & & \\ \frac{\mathrm{I}}{\Delta y^2} & B & \frac{\mathrm{I}}{\Delta y^2} & \\ & \ddots & \ddots & \ddots \\ & & \frac{\mathrm{I}}{\Delta y^2} & B \end{pmatrix} \in \mathbb{R}^{NM \times NM},\ c = -\begin{pmatrix} \frac{\alpha}{\delta x^2} \\ 0 \\ \vdots \\ \frac{\beta}{\delta x^2} \\ \vdots \\ \frac{\delta}{\Delta y^2} \\ 0 \\ \frac{\delta}{\Delta y^2} \end{pmatrix} \in \mathbb{R}^{NM}$$

and

$$B = \begin{pmatrix} \frac{-1}{\Delta y^2} + \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & & \\ \frac{1}{\Delta x^2} & \frac{-1}{\Delta y^2} + \frac{-2}{\Delta x^2} & \frac{1}{\Delta x^2} & \\ & \ddots & \ddots & \ddots \\ & & \frac{1}{\Delta x^2} & \frac{-1}{\Delta y^2} + \frac{-2}{\Delta x^2} \end{pmatrix} \in \mathbb{R}^{N \times N}.$$

For a container of width $K = 1$ and length $L = 2$, $D = 1$, $\Delta x = \frac{1}{21}$, $\Delta y = \frac{2}{40.5}$ and spray water temperature of 60°C, the heat distribution in the container is shown in Figure 7.6.

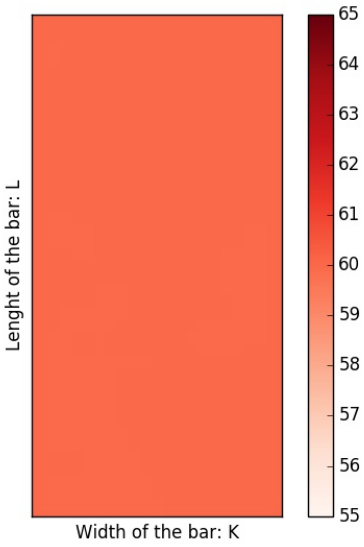


Figure 7.6: Heat distribution for a two-dimensional plate in the steady case

Dynamic Heat Equation

The dynamic heat equation in two dimensions is given by

$$\frac{\partial u(x,y,t)}{\partial t} - D\Delta u(x,y,t) = 0,\ x \in [0,K],\ y \in [0,L]. \tag{7.32}$$

The Laplacian part is discretized in both space directions as in the last section. The time derivative is discretized by the A-stable trapezoidal rule, giving the Crank-Nicolson method for the two-dimensional case:

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{D}{2}(Tu^{n+1} + Tu^n)$$

The solution of the two-dimensional non-steady heat equation is given by solving the sparse linear system

$$(\mathrm{I} - \frac{\mu}{2}T)u^{n+1} = (\mathrm{I} + \frac{\mu}{2}T)u^n - \mu c$$

with $\mu = D\Delta t$.

For a plate of width $K = 1$ and length $L = 2$, $D = 1$, $\Delta x = \frac{1}{21}$, $\Delta y = \frac{2}{40.5}$, $\Delta t = 0.01$ s and spray water temperature of 60 °C, the heat maps at different times are shown in Figure 7.7. In contrary to the one-dimesional case, the existence of a cold spot in the center of the container some centimeters above the bottom can be observed.

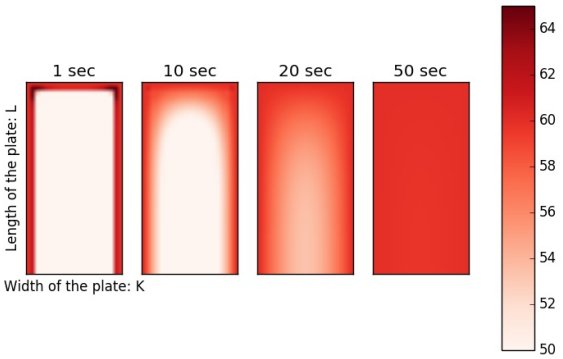


Figure 7.7: Heat distribution for a 2D-plate in the non-steady case

8 Conclusion and Future Work

The aim of this thesis is to describe and investigate two optimization methods on the background of an industrial application - the pasteurisation process of beverages.
A comparison of a gradient-based and a derivative-free method indicates that the gradient-based MMA algorithm is a serious alternative optimization method in this particular industrial context. The future work detected in this thesis includes aspects from algorithm design and development as well as the study of different model variants and optimization setups.

- *Extending MMA*: The derivative-free COBYLA algorithm supports equality constrains in contrast to the gradient-based MMA algorithm. In Chapter 6, a method to extend the MMA routine for equality constraints is presented and needs to be implemented.
- *Optimal control formulation to obtain a reference solution*: Due to construction reasons, the optimization problem is discretized with respect to the length of each heating zone. An optimal control problem can be set up without a given discretization to compare the results.
- *Optimization with dynamic scenarios*: In Chapter 4, a dynamic model is developed. This model allows to take into account stops or different speed of the conveyor belt.
- *Optimization of the overall processing time*: In this thesis, the optimization is done with respect to the PUs in the product. Instead, the time a container needs to be processed in the tunnel pasteur could be minimized, which allows to increase the amount of units to be processed per hour.
- *Optimization of the energy consumption*: Since companies want and have to reduce their energy consumption, this aspect should be taken into account and minimized.
- *Structural mechanic aspects and optimization*: The change of the spray water temperature causes stress in the container. This stress can be modeled to be incuded in the optimization problem.
- *Optimization of a more detailed model*: The model used by Krones and in this thesis ignores the convective heat transfer which takes place during the thermal treatment of the product in the tunnel pasteur. This simplification is justified by the need of a simple model with reduced computational effort in order to provide an efficient optimization. On the other hand, this simplifying assumption results in the loss of important information about the thermal processes in the product

and consequently about the pasteurisation process. Future work might follow up modeling the thermal processes including fluid flow caused by convection, see also Chapter 7. This model needs to be simplified in order to reduce the computational time and to have a competitive new model. Moreover, this model can take into account different shapes of the container.

Bibliography

[1] *Krones Pasteur*, https://www.krones.com/downloads/linaflex_en.pdf, 2016, [Online; accessed 07-March-2016].

[2] *NLopt Introduction*, http://ab-initio.mit.edu/wiki/index.php/NLopt_Introduction, 2016, [Online; accessed 12-February-2016].

[3] *NLopt Python*, http://ab-initio.mit.edu/wiki/index.php/NLopt_Python_Reference, 2016, [Online; accessed 12-February-2016].

[4] *NLopt Reference*, http://ab-initio.mit.edu/wiki/index.php/NLopt_Reference#Return_values, 2016, [Online; accessed 12-February-2016].

[5] *Patent Louis Pasteur*, http://patentimages.storage.googleapis.com/pages/US135245-0.png, 2016, [Online; accessed 9-May-2016].

[6] Shair Ahmad and Antonio Ambrosetti, *A textbook on Ordinary Differential Equations*, Springer, Cham/Heidelberg/New York/Dordrecht/London, 2014.

[7] Belinda Barnes and Glenn Robert Fulford, *Mathematical modelling with case studies. A differential approach using Maple and MATLAB (second edition)*, Taylor & Francis Group (LLC), Boca Raton, 2009.

[8] Kai-Uwe Bletzinger, *Explicit approximation of equality constraints*, Transactions on the Built Environment **2** (1993).

[9] Lars-Christer Böiers, *Mathematical Methods of Optimization*, Studentlitteratur AB, Lund, 2010.

[10] E. Dilay, Jose Viriato Coelho Vargas, Sandro Campos Amico, and Juan C. Ordonez, *Modeling, simulation and optimization of a beer pasteurization tunnel*, Journal of Food Engineering **77** (2005).

[11] Lennart Edsberg, *Introduction to computation and modeling for differential equations*, John Wiley & Sons, New Jersey, 2008.

[12] Lawrence Craig Evans, *Partial Differential Equations*, American Mathematical Society, Providence, 1998.

[13] Donald Holdsworth and Ricardo Simpson, *Thermal Procesing of Packaged Foods*, Springer Science+Business Media (LLC), New York, 2007.

[14] Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, Springer Science+Business Media (LLC), New York, 2006.

[15] Michael James David Powell, *A Direct Search Optimization Method that Models the Objective and Constraint Functions by Linear Interpolation*, Advances in Optimization and Numerical Analysis (Susana Gomez and Jean-Pierre Hennart, eds.), Kluwer Academic Publishers, Dordrecht, 1994, pp. 51–67.

[16] Krister Svanberg, *A class of globally convergent optimization methods based on conservative convex separable approximations*, SIAM Journal on Optimization **12 (2)** (2002), 555–573.

[17] EBC Technology and Engeneering Forum, *Beer Pasteurisation (Manual of good practice)*, Getränke-Fachverlag Hans Carl, Crawley, 1995.

Master's Theses in Mathematical Sciences 2016:E11
ISSN 1404-6342

LUNFNA-3021-2016

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/