Master Thesis

Shape Optimization for Injection Mold Design

carried out for the purpose of obtaining the
degree of Diplom-Ingenieur (Dipl.-Ing. or DI),

submitted at TU Wien,
Faculty of Mechanical and Industrial Engineering

by
Linus Knecht

under the supervision of
Univ.Prof.in Dr.in-Ing.in Stefanie Elgeti
Univ.Ass. Dr.-Ing. MSc Florian Zwicke
Institute of Lightweight Design and Structural Biomechanics

Affidavit

I declare in lieu of oath, that I wrote this thesis and carried out the associated research myself, using only the literature cited in this volume. If text passages from sources are used literally, they are marked as such.
I confirm that this work is original and has not been submitted for examination elsewhere, nor is it currently under consideration for a thesis elsewhere.
I acknowledge that the submitted work will be checked electronically-technically using suitable and state-of-the-art means (plagiarism detection software). On the one hand, this ensures that the submitted work was prepared according to the high-quality standards within the applicable rules to ensure good scientific practice "Code of Conduct" at the TU Wien. On the other hand, a comparison with other student theses avoids violations of my personal copyright.

Vienna, June, 2023

Signature

Acknowledgments

I would like to thank Univ.Prof.in Dr.-Ing.in Stefanie Elgeti and the whole staff at the Institute of Lightweigt Design and Structural Biomechanics, who gave me the opportunity to write my Master's Thesis at this institute.
I would like to express my special thanks to my supervisor, Univ.Ass. Dr.-Ing. Florian Zwicke who supported me constantly and helped me to achieve the result of this thesis by professional discussions and helpful suggestions.
My biggest thanks go to my family and friends, who supported me during my studies and my Master's Thesis.

Contents

Abstract

A method for the shape optimization of a cavity for the injection molding process is implemented and tested. This is necessary because shrinkage and warpage occur during the cooling process. As a result, the shapes of the cavity and the molding differ. Since the shape of the molding depends directly on the cavity, an environment is implemented in PYTHON which allows to determine an optimized shape of the cavity.

The process of injection molding is modelled using the finite element method. The process before ejection is simulated with aid of help of a time-dependent thermal conductivity model. Shrinkage and warpage that occur during cooling after ejection are described by a non-linear elasticity equation which also takes thermoelastic material behavior into account.

These simulations are integrated into an optimization algorithm which utilizes free-form deformation to change the shape of the cavity. The aim of the optimization is to achieve the smallest possible deviation between the geometry of the molding and a reference geometry. For this purpose, an objective function is introduced, which defines a scalar value for the shape deviation.

The framework is examined in 3D in relation to the impact of the choice of different optimization algorithms, the influence of its design parameters, their sensitivities and the impact of an initial guess. Finally, a real-world application case is optimized with the presented method.

Kurzfassung

Eine Methode zur Formoptimierung einer Kavitätsgeometrie für den Kunststoffspritzguss wird implementiert und getestet. Dies ist notwendig, da es während des Abkühlungsvorgangs beim Kunststoffspritzgussverfahren zu Schwindungen und Verzug im Bauteil kommt, was zur Folge hat, dass sich die Formen von Kavität und Werkstück unterscheiden. Da die Werkstückform direkt von der Kavität abhängt, soll eine Umgebung in PYTHON geschaffen werden, die es erlaubt eine geeignete Form der Kavität zu ermitteln.

Der Prozess des Kunststoffspritzgusses wird mithilfe der Finiten Elemente Methode (FEM) modelliert. Dabei wird der Prozess vor dem Auswurf mithilfe einer zeitabhängigen Wärmeleitungsgleichung simuliert. Schwindung und Verzug während des Abkühlens nach dem Auswurf wird mit einer nicht-linearen Elastizitätsgleichung beschrieben, welche zusätzlich das thermoelastische Materialverhalten berücksichtigt.

Diese Simulation sind in einen Optimierungsalgorithmus eingebunden, welcher mithilfe einer Free-Form-Deformation die Kavitätsgeometrie verändert. Ziel der Optimierung ist dabei eine möglichst kleine Abweichung zwischen der Geometrie des Werkstücks und einer Referenzgeometrie. Hierfür wird eine Zielfunktion eingeführt, welche einen skalaren Wert für die Formabweichung errechnet.

Die Umgebung zur Formoptimierung der Kavitätsgeometrie für den Kunststoffspritzguss wird in 3D untersucht. Dabei wird die Auswahl verschiedener Optimierungsalgorithmen, der Einfluss der Design Variablen, der Sensitivtäten sowie einer passenden Initialisierung untersucht. Schließlich wird ein realer Anwendungsfall mit der vorgestellten Methode optimiert.

List of Figures

List of Tables

List of Symbols and Abbreviations

Greek Symbols

α_{th}	Thermal Modulus
γ	Thermal Expansion Factor
Γ	Boundary of a Domain
$\delta(\mathbf{A},\mathbf{B})$	Hausdorff Distance between Point Clouds $\mathbf{A}$ and B
Δ	Laplace Operator
η	Parametric Coordinate
$\mathcal{H}$	Knot Vector
κ	Thermal Diffusivity
λ	Lame Constant
μ	Lame Constant
ν	Poisson Number
ξ	Parametric Coordinate
Ξ	Knot Vector
Π	Energy Functional
ρ	Density
Ω	Domain
$\tilde{\Omega}$	Initial Domain
$\overline{\Omega}$	Result Domain
$\hat{\Omega}$	Target Domain
Ω^{ffd}	Free-Form Deformation Domain
∇	Nabla Operator

Latin Symbols

$\mathbf{B_{i,j}}$	Control Net
c_p	heat capacity
$\mathbf{C}$	Cauchy-Green Deformation Tensor
d	Dimension
D_J	Objective Function Value
$\mathbf{E}$	Green-St. Venant Deformation Tensor
$\mathbf{F}$	Formal Deformation Gradient Tensor
$\mathbf{I}$	Identity Matrix
$J(\mathbf{x})$	Objective Function
k	Thermal Conductivity
k_{CP}	Control Point Index
L_p	Metric with Degree p

$\mathbf{m}$	Mesh
n	Function Evaluations
$N_{i,p}$	Basis Function for Spatial Dimension i and Degree p
N_k	Interpolation Function
N_{CP}	Total Number of Control Points
$\mathbf{P_{k_{CP}}}$	Control Point with index k_{CP}
$\mathbf{q}$	Local Heat Flux
r	Threshold for the Sensitivity
$Res(v, T)$	Residual
$\mathbf{S}$	Basis-Spline
T	Temperature
tr	Trace of a Matrix
$S_i^{k_{CP}}$	Sensitivity of a Control Point in Spatial Direction i
$S(v, T)$	Dirichlet Boundary Condition
$\mathbf{u}$	Displacement Vector
$v(\mathbf{x})$	Test Function
W	Strain-Energy Density
$\mathbf{x}$	Coordinates Vector

Abbreviations

CP	Control Point
DP	Design Parameter
DoF	Degree of Freedom
FEM	Finite Element Method
FFD	Free Form Deformation
IG	Initial Guess
IMO	Injection Molding Optimizer

1. Introduction

One of the major processes of producing plastic parts is the injection molding process. Typical examples for injection molded parts are mobile phone or computer cases, lighting parts in the automotive sector or small plastic parts like bottle caps or cable clamps.
During injection molding, hot and liquid plastic is injected into a cavity shape, and eventually, the part will be ejected and fully solidifies. However, due to shrinkage and warpage during the cooling process, the cavity shape can deviate from the shape of the mold (see Figure 1). Another challenge in producing plastic components, is the trend of producing biodegradable plastics. This includes taking different material properties and deformation behavior into account.

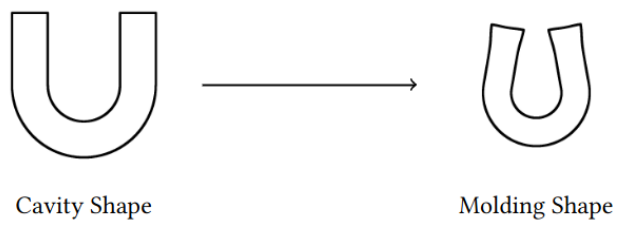


Figure 1: Difference between the cavity shape and the molding shape [41].

Facing the problem of finding the desired cavity shape in order to achieve an optimal molding shape, computational methods can be of use. Typical approaches in engineering are the simulation of physical processes with the finite element method (FEM). However, since a single forward simulation only provides information of the current configuration, it is not sufficient to determine the optimal shape. Instead, one needs to solve the inverse problem, e.g. via optimization procedures. One approach to address this problem has been introduced in [41] and is based on directly solving the inverse problem. The method is computationally very efficient, but suffers from the drawback that it considers the cooling process as steady-state, i.e. infinitely slow [42]. Hence, as an alternative approach, we investigate automatized shape optimization. The second approach proposed in this work, is the combination of FEM with a mathematical shape optimization. This idea is already applied e.g. in [14] for the process of plastic extrusion. Unlike the first approach, this allows a transient consideration of the cooling process. Further advantages of shape optimization are the option to include plastic deformation as well as optimizing additional features, such as residual stresses or material parameters.

Shape optimization approaches can be found in many fields of engineering, from aerodynamics to viscous flow analysis [28, 21]. An integral part of shape optimization is the method that is responsible for the shape deformation. In this work, we employ free-form deformation (FFD) [30]. Applying FFD in combination with FEM, the finite element mesh serves as the basis for both solving the governing equations and the geometry representation. In order to deform the finite element mesh smoothly to yield geometry updates, the finite element mesh is embedded in a box spline, which can be easily deformed by using relatively few geometric parameters. In this case, these parameters are the positions of the control points.

Another important choice in shape optimization is the definition of an optimization goal. Recent studies in the field of shape optimization for injection molding processes focus on different optimization problems. In [40], the overall stresses within the molding are optimized, while in [32] the reduction of material waste is aimed at. However, this work focuses on the optimization of the cavity shape (see Figure 2), such that the desired molding shape is returned after the shrinkage and warpage process.

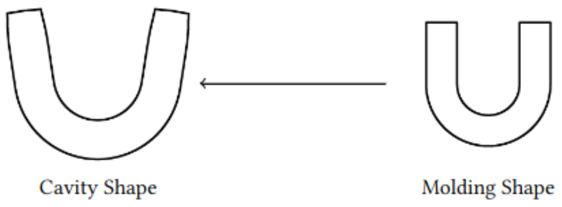


Figure 2: The molding shape is specified. With shape optimization, the cavity shape can be determined [41].

In this work, a PYTHON-based framework was implemented that combines the simulation of the injection molding process with a shape optimization. An overview of the injection molding process, the finite element method, shape optimization as well as the used software tools in this work are described in Chapter 2. Chapter 3 focuses on the methods for the simulation and optimization, including the definition of the objective function and the free-form deformation. Further, the injection molding optimization framework is proposed in this chapter. Subsequently, the framework is tested in Chapter 4 regarding different optimization algorithms, the set of design variables, the proposed sensitivity analysis, a proposal of an initial guess and two different objective functions. The findings from this chapter are discussed in Chapter 5. Combining these results, in Chapter 6 the optimization of a cable clamp is shown, in order to give an example for a real-world application of the proposed method. Finally, Chapter 7 sums up the results of this work, and ideas for future work are proposed.

2. Theoretical Foundations

In this chapter, the injection molding process (see Section 2.1) and the finite element method (see Section 2.2) are described briefly. Further, the fundamental concept of shape optimization (see Section 2.3) including some useful optimization algorithms are depicted. Finally, the software tools used in this work are described in Section 2.4.

2.1. Injection Molding Process

Injection molding is a manufacturing process for producing plastic parts. Molten plastic is injected into a cavity form, where it cools down and forms the product [4]. Figure 3 shows the process of injection molding in a scheme. In the following, the process steps are briefly explained [41].

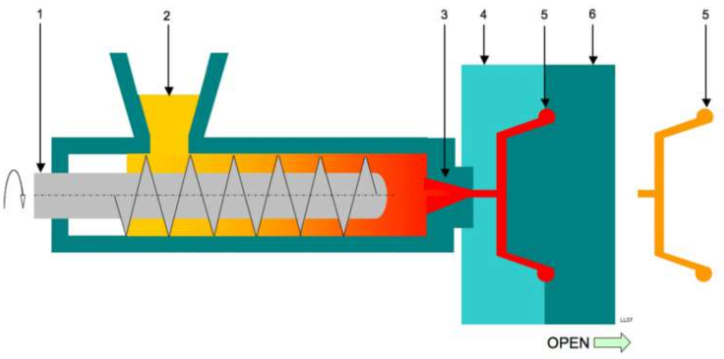


Figure 3: Scheme of the injection molding machine. The plasticizing mainly includes the screw (1), the hopper (2) and the nozzle (3). The mold is formed by two halves (4) and (6). At the end of the process, the part (5) can be ejected [33].

Plasticizing
At first, the material (e.g. plastic pellets, solid granules) is filled into the plasticizing unit through the hopper. By the rotation of the screw, the granules are transported forward and compressed at the front of the screw. Both friction and the heated walls of the plasticizing unit cause the material to heat up and melt.

Filling
Once enough molten plastic has accumulated at the front of the plasticizing unit, the nozzle opens, and the molten plastic is injected into the mold. In order to press the plastic out of the cylinder, the screw is pushed forwards. As soon as the plastic melt comes into contact with the mold, it starts to solidify. The molds walls are actively cooled down, in order to ensure fast solidification.

Packing and Cooling
Due to shrinkage during material cooling, more plastic melt must be injected into the mold for as long as possible. As such, a constant pressure is maintained on the melt. The nozzle can be closed afterwards, and the plasticizing unit is ready to use for the next melting process.

Ejection
The partially solidified molding is ejected by the opening of the mold. The workpiece is not fully cooled down yet and will shrink and warp further during the continued cool down. The part is either ejected or has to be extracted manually.

2.2. Finite Element Method

Numerical models of material behavior are often based on partial differential equations. Since the finding of an analytical solution of a partial differential equation ranges from complex to impossible, a numerical solution is required.
The main idea of a numerical scheme is to search for a solution in a discrete or finite-dimensional space. The finite element method is one such scheme.
The main idea in FEM is to split the computational domain into a finite number of subdomains. These subdomains mostly have the form of simple geometric shapes, such as triangles and rectangles in 2D or prismatic and hexahedral elements in 3D.
In FEM, values of the unknown function (e.g. temperature) are computed at so-called nodes, the intersection corners of elements. In between the nodes, the values are interpolated using e.g. Lagrange polynomials or B-splines. Inserting this formulation into form of the underlying partial differential equation generates a linear system for the unknown function values. These are then determined using appropriate solution algorithms for solving (nonlinear) systems of equations [12].

2.2.1. Geometry and Topology

When subdividing a computational domain into a set of non-overlapping elements, one generates a so-called mesh. This mesh is fully defined by two properties: (1) The position of the nodes comprising the mesh, and (2) the connectivity between nodes [13]. The nodal positions are stored in a file named *Geometry*, whereas the connectivity is stored in the *Topolgy* file.

2.2.2. Element Type

Before the FEM calculation can be started, a suitable element type must be selected. There are a number of possibilities, depending on the number of nodes of the reference element. Second-order hexahedral elements are the most common used elements for structural simulations. However, automatic meshing produces tetrahedral meshes much better than hexahedral meshes. Comparative studies have demonstrated that second-order tetrahedral meshes show good results compared to hexahedral meshes of the same order [7]. Only for elasto-plastic deformation, the hexahedral mesh outperforms the tetrahedral mesh [3].
This work focuses mainly on the implementation of second-order hexahedral elements. Yet, it is fully applicable for second-order tetrahedral meshes as seen in Section 6.

2.3. Shape Optimization

2.3.1. Overview

Generally, optimization (or mathematical programming) aims at finding one or more optimal parameters for a system. While optimization is used in various fields, such as physics, biology, engineering, economics and business, the underlying mathematical principles are identical [39]. Usually, shape optimization can be understood as a minimization problem. To minimize a function or a system, one must find the minimum of the objective function $J(\mathbf{d})$:

$$\min J(\mathbf{d}). \tag{1}$$

The objective function depends on the vector of the design parameters $\mathbf{d}$. Discussing the theory of minimization is sufficient, since $\max J(\mathbf{d}) = -\min(-J(\mathbf{d}))$. Minimization problems often include constraints or inequalities that need to be satisfied [29].

In engineering applications, structural optimization has become of great importance. For example, construction work, automobile, or airplane structures can be optimized for improved strength or stiffness properties and reduced weight or cost. Structural optimization can be divided into two areas of research, the optimization of shape and the optimization of the topology of the structural configuration [16].

Whereas topology optimization focuses on the distribution of the material within the finite element model, shape optimization focuses on the geometry of the structure and requires redefinition of the finite element mesh [16]. The design parameters are somehow connected to the shape of the geometry. Changes within the topology, such as adding or removing parts, are not allowed [16, 29].

Usually, structural optimization follows a certain scheme, as illustrated in Figure 4. At first, the model is initialized with input parameters from an *Initial Draft*, which is used to compute a solution of the *Analytical Model*. By *Evaluating* the objective function, a scalar value can be transferred to the *Optimization Algorithm*. The latter determines, based on constraints and other restrictions, the new design parameters. If a point is reached that satisfies the optimization criteria, the algorithm stops and the optimization is successful [29].

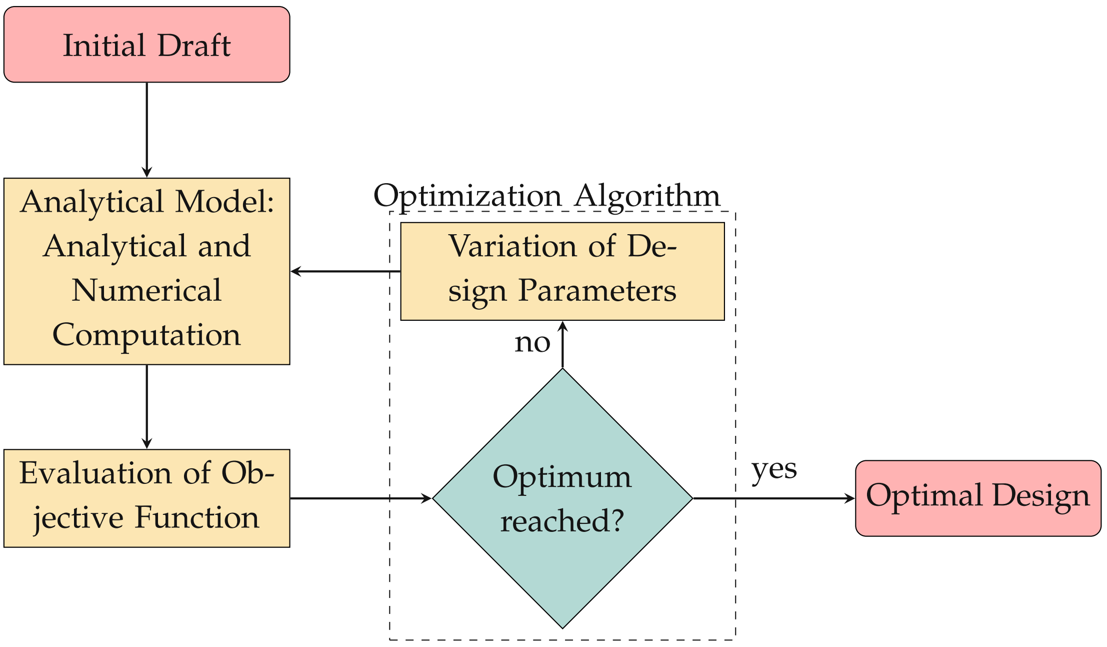

Initial Draft
Analytical Model: Analytical and Numerical Computation
Optimization Algorithm
Variation of Design Parameters
no
Evaluation of Objective Function
Optimum reached?
yes
Optimal Design

Figure 4: General shape optimization loop. Adapted and inspired by [29].

2.3.2. Optimization Algorithms

As mentioned above, the goal of the optimization is the minimization of a function $J(\mathbf{x})$. In numerical computing, there is need of iterative methods to yield the minimum of a function. Quite popular minimization algorithms are e.g. the *gradient descent algorithm* or *Newton's method*. The *gradient descent algorithm* needs first order derivative information, whereas *Newton's method* needs both first and second order derivative information. However, there are optimization problems, upcoming from scientific, engineering or artificial intelligence applications, where only the output of a simulation is known. Due to the fact, that the objective function for lacks a derivative, a derivative-free optimization or an approximation of the derivative is required [19]. In the following, an overview of the optimization algorithms used in this work is given.

BFGS

BFGS is the acronym for the Broyden-Fletcher-Goldfarb-Shanno algorithm and is a gradient-based iterative algorithm for solving unconstrained nonlinear optimization problems. BFGS usually needs the gradient of a function, and is therefore no *derivative-free optimization algorithm*. However, if no gradient is available, it can be approximated by the difference quotient. BFGS is a quasi-Newton method, in particular the Hessian matrix is approximated at every iteration. The main idea is the minimization of difference between the current and the last approximation of the Hessian matrix. This is done by comparing the inversions of the Hessian matrix approximations with the Frobenius norm. BFGS uses the line-search method in order to find the minimum of the problem [15].

BOBYQA

BOBYQA stands for Bounded Optimization BY Quadratic Approximation and was invented by M. Powell in 2009 and tackles, like all of Powell's algorithms, nonlinear problems. The algorithm is a trust region approach trying to yield a quadratic model of the objective function by interpolation. As there exists a Hessian matrix of any quadratic function, the algorithm is somehow similar to the BFGS algorithm. BOBYQA is minimizing the Frobenius norm of the comparison between the Hessian matrices of the model and the function by a similar version of the Brodyn formula. So the main difference between BFGS and BOBYQA is the search strategy. Note, that BOBYQA works only with bounded constraints [25].

LINCOA

The LINCOA solves linear constraint optimization problems. Like BOBYQA, it uses a quadratic approximation of the trust-region model and minimizing the Frobenius norm. It is an adaption to BOBYQA, as it also solves linear constraint problems [26].

COBYLA

COBYLA stands for Constrained Optimization BY Linear Approximation and was invented by M. Powell in 1994. COBYLA is a direct search optimization algorithm which uses linear models built by approximation for nonlinear constraint problems. The algorithm constructs a linear model, by interpolating the vertices of a simplex. While minimizing the problem with the Simplex-Method, the algorithm does also propose a new set of simplices iteratively. Since this algorithm uses linear programming, it is different to the algorithms mentioned above [24].

2.4. Software Tools

Nutils

All simulations described in this work are performed with the simulation software NUTILS developed by Evalf [35]. NUTILS is a free and open-source library for FEM computations written in the programming language PYTHON [34]. Nutils is a well-suited platform for numerical science, since it supports Isogeometric Analysis, the Finite Cell Method (FCM), multi-physics, mixed methods, and hierarchical refinement.

Gustaf

GUSTAF is a PYTHON library to process and visualize numerical-analysis-geometries and was developed by Tataratat at the Institute of Lightweight and Structural Biomechanics (TU Wien). It supports linear elements and is used especially for FEM and Isogeometric Analysis. In this work, GUSTAF is used to perform the free-form deformation as well as mesh generation and visualizations.

PDFO

Powell's Derivative-Free Optimization solvers (PDFO) is a cross-platform package providing interfaces for using Powell's derivative-free optimization solvers, which were originally implemented in Fortran 77. PDFO wraps the original code from Powell into the more user-friendly PYTHON environment [27].

SciPy

In the PYTHON library SCIPY [37] are many algorithms for optimization, integration, interpolation and more mathematical problems implemented. SCIPY provides the BFGS algorithm, which is described above.

PCU

POINT CLOUD UTILS or PCU is a PYTHON library for manipulating and processing 3D point clouds and meshes [38]. It provides the implementation of the *Hausdorff Distance*, used in this work.

3. Methods

In this chapter, the simulation model for the injection molding process and important terminology is introduced. Further, the mathematical formulation of the shape optimization framework, including the definition of the objective function and the free-form deformation, will be explained. Finally, the implemented injection molding optimization (IMO) framework is depicted.

3.1. Simulation

This section contains the necessary equations to model the process of injection molding in a simplified manner. To keep the model simple, and therefore the computational effort low, simplifying assumptions are made. For example, viscoelasticity, crystallization as well as the in-flow behavior are neglected here. We divide the entire injection molding process into two parts. The process before the ejection is described by the heat conduction equation, while the part after the ejection considers the nonlinear elasticity of the polymer. The heat simulation, as well as, the shrinkage and warpage simulation are described in the following sections. In particular, we state the governing equations as well as the discretization method. For more detailed information about the discretization method, refer to [12, 13].

3.1.1. Heat Simulation

As mentioned above, the process before the ejection is modelled by the heat conduction equation. The transient simulation for this process is described in this section.

Governing Equations

Fourier's law of thermal conduction reads

$$\mathbf{q} = -k\nabla T, \tag{2}$$

where $\mathbf{q}$ is the local heat flux density, k the thermal conductivity and ∇T the temperature gradient. Considering a small volume, the first law of thermodynamics yields

$$\frac{\partial u}{\partial t} + \nabla \mathbf{q} = 0. \tag{3}$$

Here, u describes the internal energy per unit volume of the system, t the time and $\nabla \mathbf{q}$ the change of the heat flux. The differential equation for the internal energy is defined as

$$\frac{\partial u}{\partial t} = c_p \rho \frac{\partial T}{\partial t}, \tag{4}$$

where c_p is the heat capacity under constant pressure and ρ the density. For a homogeneous medium, the heat conduction equation then yields

$$\frac{\partial T}{\partial t} - \kappa \Delta T = 0. \tag{5}$$

The thermal diffusivity κ is defined as

$$\kappa = \frac{k}{c_p \rho}. \tag{6}$$

Spatial Discretization

The space-dependent part of the heat equation (see Equation 5) is discretized in this section. The weak formulation of the heat equation yields

$$\int_\Omega \kappa \nabla T(\mathbf{x}) \nabla v(\mathbf{x}) d\Omega = 0, \tag{7}$$

where $v(\mathbf{x})$ are the test functions of the problem and Ω the domain. We require a discrete solution variable to obtain a numerical formulation of the problem. Therefore, by using the interpolation functions N_k for a node k, we write

$$T(\mathbf{x}) = \sum_{i=1}^{k} N_k \tilde{T}_k \tag{8}$$

and

$$v(\mathbf{x}) = \sum_{i=1}^{k} N_k \tilde{v}_k, \tag{9}$$

where $\tilde{T}_k$ and $\tilde{v}_k$ describe the nodal values. After partial integration, one obtains the residual for the domain Ω

$$Res(v, T) := \int_\Omega \kappa \nabla T(x) \nabla v(x) d\Omega \overset{!}{=} 0. \tag{10}$$

After numerical integration, a linear system of equations is obtained, which needs to be solved. The Dirichlet boundary condition can be defined as

$$S(v,T) := \int_\Gamma (T - T_{Dirichlet})^2 d\Gamma \overset{!}{=} 0. \tag{11}$$

By seperatively minimizing $S(v,T)$ from the heat equation, we find the space constraints for the Dirichlet boundary condition to obtain the boundary values for Equation 10. This yields the condition $T = T_{Dirichlet}$ on Γ, but needs to be implemented in NUTILS as defined in Equation 11. Note, that Γ defines the boundary of the domain.

Temporal Discretization

In addition to discretizing in space, a temporal discretization is needed in order to solve the time-dependent heat equation (see Equation 5). In the work at hand, the implicit Euler method is used for solving the heat equation. For a time step n, the discretization scheme yields

$$\mathbf{T}^{n+1} = \mathbf{T}^n + \Delta t \mathrm{f}(\mathbf{T}^{n+1}, t^{n+1}), \tag{12}$$

with Δt being the time step size and f the derivative of the function $\mathbf{T}$ with respect to the time t. As this method is an implicit time scheme and one finds the variable $\mathbf{T}^{n+1}$ on both sides of the equation, which leads to a system of equations for every time step.

3.1.2. Shrinkage and Warpage Simulation

The process after the ejection of the workpiece can be simulated with the shrinkage and warpage simulation. In order to simplify this part of the simulation, we assume that the material has solidified completely at the time of ejection. This way we can use a stationary formulation of thermoelasticity. Hyperelastic material models for non-linear elasticity are widely used in isothermal settings. Since thermal expansion and contraction plays a crucial role in this process, we need to make some adjustments to the isothermal formulations.

Governing Equations

The strain-energy density function for the Saint-Venant-Kirchhoff model is defined as

$$W(\mathbf{E}) = \frac{\lambda}{2}[\mathrm{tr}(\mathbf{E})]^2 + \mu \mathrm{tr}(\mathbf{E}^2), \tag{13}$$

where λ and μ are the Lamé parameters and $\mathbf{E}$ is the Green-St. Venant deformation tensor [6]. It is defined as

$$\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I}), \tag{14}$$

with $\mathbf{I}$ being the Identity Matrix and

$$\mathbf{C} = \tilde{\mathbf{F}}^\top \tilde{\mathbf{F}} \tag{15}$$

being the right Cauchy-Green deformation tensor [6, 17]. We define the formal deformation gradient tensor with respect to a displacement $\mathbf{u}$ as

$$\mathbf{F} = \nabla \mathbf{u} + \mathbf{I}. \tag{16}$$

Taking thermal expansion of the material into account, the deformation gradient tensor is defined as

$$\tilde{\mathbf{F}} = \gamma^{-\frac{1}{d}} \mathbf{F}, \tag{17}$$

where d is the spatial dimension. For more detailed information about these equations, refer to [6, 17, 41].
The thermal expansion factor γ can be linearized as

$$\gamma(T) = 1 + \alpha_{th}(T - T_0), \tag{18}$$

with the thermal modulus α_{th} [20]. Here, T is the temperature and T_0 the initial temperature.

Discretization

The displacements $\mathbf{u}$, which occur during this process, can be approximated by

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{k} v_k(\mathbf{x}) \tilde{\mathbf{u}}_k(\mathbf{x}), \tag{19}$$

where v_k are the test functions of the problem. The energy functional for a domain Ω is defined as

$$\Pi := \int_{\Omega} W(\mathbf{E}) d\Omega, \tag{20}$$

with $W(\mathbf{E})$ being the strain-energy-density function for the Saint-Venant-Kirchhoff model as formulated in Equation 13). This functional can now be minimized in order to obtain the solution for the displacement $\mathbf{u}$.

3.2. Domain and Mesh Terminology

For a better comparison, we need to introduce different states of the domain as mentioned in [41]. First, we define an *initial domain*

$$\tilde{\Omega} \subset \mathbb{R}^d, \tag{21}$$

which will be the start domain of any simulation. The dimension of the domain is given by the exponent d. The corresponding coordinates of this domain would be

$$\tilde{\mathbf{x}} \in \tilde{\Omega}. \tag{22}$$

As we have seen in the previous chapter, the goal of our simulation model is to obtain the displacement field $\overline{\mathbf{u}} : \tilde{\Omega} \to \mathbb{R}^d$. The result is called the *result domain* and can be described through

$$\overline{\Omega} := \{\overline{\mathbf{x}} \in \mathbb{R}^d : \overline{\mathbf{x}} = \tilde{\mathbf{x}} + \overline{\mathbf{u}}(\tilde{\mathbf{x}}), \tilde{\mathbf{x}} \in \tilde{\Omega}\}. \tag{23}$$

By knowing the displacement field $\overline{\mathbf{u}}$, we can simply determine the resulting state by adding all the coordinates $\tilde{\mathbf{u}}$ of the *initial domain* with the displacement field $\overline{\mathbf{u}}$.
The shape of the *result domain* is the shape, which we want to optimize, in order to be as close as possible to a *target domain*

$$\hat{\Omega} \subset \mathbb{R}^d. \tag{24}$$

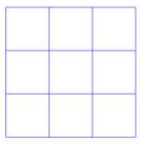
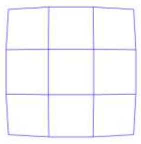
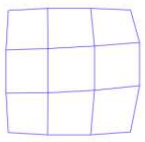
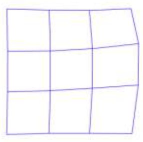
This shape is of importance for formulating the objective function (see Section 3.3.1). For the sake of clarity, it has to be mentioned that we introduce a fourth state of the domain, the *free-form deformation domain*

$$\Omega^{ffd} \subset \mathbb{R}^d. \tag{25}$$

This state is discussed more detailed in Section 3.3.2.

The difference between a domain and a mesh has to be highlighted. Whilst the domain is a design space (e.g. a surface or volume), the mesh is the discretized form of the domain. Therefore, the domain is approximated by a set of smaller and simpler elements. However, the state of the mesh corresponds to the state of the domains as explained above. A visualization of the different states of a mesh is shown in Table 2.

Table 2: Definition of mesh terms.

	Name	Description
	$\hat{\mathbf{m}}$	The reference mesh is the targeted shape of the iteration process. It is required for determining the objective function value.
	$\tilde{\mathbf{m}}$	The initial mesh is the input mesh for the optimization loop. It can be derived from a guess or estimated (see Section 3.4.4).
	$\mathbf{m}^{\mathbf{ffd}}$	The FFD mesh is the start mesh for the simulation. It is the deformed mesh based on the variation of the input parameters (see Section 3.3.2). It is further used as the current cavity shape.
	$\overline{\mathbf{m}}$	The simulated mesh or result mesh is the resulting mesh after the simulation is performed. It is the shape, which has to be optimized.

3.3. Optimization Strategy

This section provides the outline for the shape optimization framework. Firstly, two potential objective functions are introduced, followed by an description of free-form deformation, which is essential for updating the initial geometry. Subsequently, the optimization strategy is presented.

3.3.1. Objective Function

Describing an objective function $J(\mathbf{x})$ gives us the opportunity to express an abstract problem, such as determining the difference between the result shape $\overline{\Omega}$ and the targeted shape $\hat{\Omega}$, by means of a scalar value. Therefore, the objective function J is a function of the type

$$J : \overline{\mathbf{x}}, \hat{\mathbf{x}} \rightarrow \mathbb{R}, \tag{26}$$

where $\overline{\mathrm{x}}$ and x respectively describe the associated coordinates of the shapes. In the following, two different objective functions will be introduced.

Distance Function

The L_p-metric is a commonly used formulation measuring similarities of shapes. For two points x,y in $\mathbb{R}^k$, the L_p-distance is defined as

$$L_p(x, y) = \left(\sum_{i=0}^{k} (\mid x_i - y_i \mid)^p \right)^{\frac{1}{p}}. \tag{27}$$

For any $p \geq 1$, $L_p(x, y)$ defines a metric. If $p = 2$, one obtains the *Euclidean Distance*, which returns the distance between two points x,y in the Euclidean Space [36].

With $p = 2$, we obtain the Euclidean distance of two corresponding points. Each point of $\overline{\mathrm{x}}_i$ can be uniquely assigned to a point of $\hat{\mathrm{x}}_i$.
We can now integrate over the whole boundary $\overline{\Gamma}$ of the *result domain*, in order to sum up these distances. The objective function is then defined as

$$J(\overline{\mathbf{x}}, \hat{\mathbf{x}}) := \frac{\displaystyle\int_{\overline{\Gamma}} L_2(\overline{\mathbf{x}}_i, \hat{\mathbf{x}}_i) d\overline{\Gamma}}{\displaystyle\int_{\overline{\Gamma}} d\overline{\Gamma}}. \tag{28}$$

The *Distance Function* can by understand as a summation of all distances over the boundary of the domain and is normalized by the domain boundary.

Hausdorff Distance

Another frequently used distance measurement formulation applied to point clouds is the *Hausdorff Distance*. Furthermore, it can be applied to point clouds containing different numbers of points [36]. For two point clouds $\mathbf{A}$ and $\mathbf{B}$ in $\mathbb{R}^2$ or $\mathbb{R}^3$, the *directed Hausdorff Distance* is

$$\vec{\delta}(\mathbf{A}, \mathbf{B}) = \sup_{a \in A} \inf_{b \in B} \|a - b\|, \tag{29}$$

where $\|.\|$ is the Euclidean metric [2]. Here, a and b are members of the point clouds $\mathbf{A}$ and $\mathbf{B}$, respectively. The supremum defines the lowest upper bound, and the infimum the highest lower bound, respectively. The term $\inf_{b \in B} \|a - b\|$ describes the (smallest) distance from a point a to the point cloud $\mathbf{B}$. This means, $\vec{\delta}(\mathbf{A}, \mathbf{B})$ is defined as the lowest upper bound (supremum) over all points in $\mathbf{A}$ of the distances to $\mathbf{B}$ [36]. In Figure 5, the directed *Hausdorff Distance* from $\mathbf{A}$ to $\mathbf{B}$ is illustrated with a dashed line.

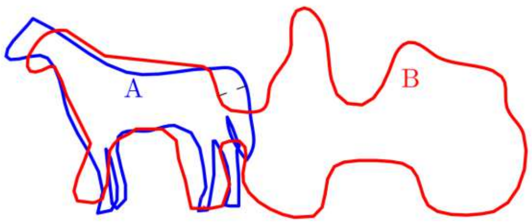


Figure 5: Illustration of the *Hausdorff Distance* between two sets A and B [2].

For a full comparison of both shapes, one uses the *Hausdorff Distance* given by

$$\delta(A, B) = \max \left\{ \vec{\delta}(A, B), \vec{\delta}(B, A) \right\}. \tag{30}$$

The objective function based on the *Hausdorff Distance* is defined as

$$J(\overline{\mathbf{x}}, \hat{\mathbf{x}}) := \delta(\overline{\mathbf{x}}, \hat{\mathbf{x}}). \tag{31}$$

3.3.2. Free-Form Deformation

Free-form deformation (FFD) was first described by Sederberg et al. [30] and has been widely used in the context of solid deformation of geometric models. In FFD, a polynomial box-spline is constructed around a solid (in this case a mesh) to be deformed. By moving its control points in the physical space, one obtains a deformed solid. In this work, FFD is performed by the use of so-called B-splines (basis splines). They will be explained briefly in the following section.

B-Splines

In terms of simplicity, only 2-dimensional splines will be observed in the following. A more detailed description about splines and solid B-splines can be found in [9]. Assuming an arbitrary control net (or control mesh) $\mathbf{B}_{i,j}$ with $i = 1, 2, ..., I$ and $j = 1, 2, ..., J$ and the knot vectors $\Xi = \{\xi_1, \xi_2, ..., \xi_{n+p+1}\}$ and $\mathcal{H} = \{\eta_1, \eta_2, ..., \eta_{n+p+1}\}$, a 2D-surface B-spline (basis spline) is given by

$$\mathbf{S}(\xi, \eta) = \sum_{i}^{m} \sum_{j}^{n} N_{i,p}(\xi) N_{j,q}(\eta) \mathbf{B}_{i,j}. \tag{32}$$

Hereby, $N_{i,p}(\xi)$ and $N_{j,q}(\eta)$ are the B-spline basis functions, and the indices p and q describe the polynomial order of the spline. The vectors ξ and η denote the parametric coordinates.

Basis Function

The basis functions for the B-Splines are generated by using the *Cox-de Boor formulation* [10]. For a specific knot vector Ξ and for $p = 0$ the basis functions are defined as

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1}, \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

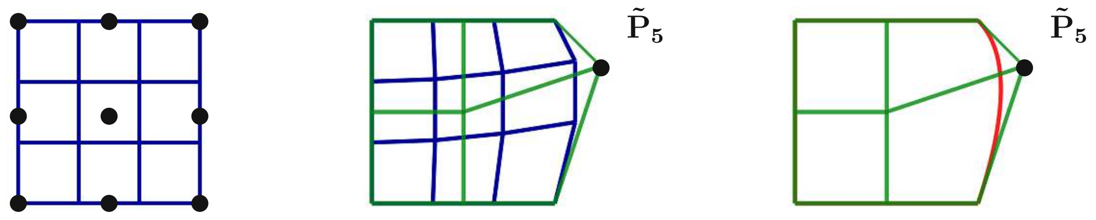
and for $p = 1, 2, 3, ...$ the basis functions are defined recursively as

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) - \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \tag{34}$$

Solid Deformation

A deformation of the parametric grid can be achieved by manipulation or translation of the respective control points, and hence the reference mesh. Subsequently, the physical coordinates of the deformed body can be recalculated. Therefore it is sufficient to move the unique control points $\mathbf{P}_{\mathbf{k_{CP}}}$ which are contained within the control net $\mathbf{B}_{\mathbf{i,j}}$. We define a set of control points $\{\mathbf{P}_0, \mathbf{P}_1, ..., \mathbf{P}\}$ with $\mathbf{P}_{\mathbf{k_{CP}}} \in \mathbb{R}^d$ and $0 \leq k_{CP} \leq N_{CP} = I \cdot J$ denoting the total number of control points, which refers to $\mathbf{B}_{i,j}$ in Equation 32. In terms of simplicity, we address the set of control points by a single index k_{CP}.

An examplary free-form deformation of a 2D-mesh incorporating 9 elements is illustrated in Figure 6. Thereby, by embedding the initial mesh (see Figure 6a) and a control



(a) Original Mesh with 9 control points
(b) Deformed Mesh
(c) Deformed Spline

Figure 6: Deformation of the mesh (blue) due to free form deformation. The spline is illustrated in red, and the control point mesh in green.

mesh consisting of $N_{CP} = 9$ control points, the initial shape can be modified by means of a FFD. In particular, the control point $\mathbf{P}_5 = (2, 1)$ is shifted to $\tilde{\mathbf{P}}_\mathbf{5} = (2.5, 1.5)$, as seen in Figure 6b. Through the displacement of the control point, the surface spline (red) is deformed as illustrated in Figure 6c. Therefore, by applying FFD, a deformation of the original mesh can be achieved by translating the control points of a surface spline.

Design Parameters

A design parameter corresponds to the spatial component $i = x, y, z$ of a single control point $\mathbf{P}_{\mathbf{k_{CP}}}$. Therefore, the degree of freedom of a control point $\mathbf{P}_{\mathbf{k_{CP}}}$ indicates the number of spatial axes along which it can be moved. It may be necessary to restrict movement along certain coordinate axes by reducing the degree of freedom of a control point.
Thus, from one single control point we obtain as many design parameters (DP) for the objective function as specified by the degree of freedom. In other words, a single design parameter describes the translation of a control point along a specific coordinate axis.

3.3.3. Problem Formulation

In order to identify the optimal shape of the mold, we can formulate the shape optimization problem as

$$\text{Find} \arg\min J(\overline{\mathbf{x}}(\mathbf{P}_{\mathbf{k_{CP}}}), \hat{\mathbf{x}}) \tag{35}$$

with $\mathbf{P}_{k_{CP}} \in \mathbb{R}^d$ for $0 \leq k_{CP} \leq N_{CP}$. The overall goal of the optimization loop is minimizing the difference between the reference shape and the shape of the configuration after the cooling and shrinkage process completed. Therefore, a free-form deformation of the initial mesh is necessary, as well as the simulation starting from the deformed body Ω^{ffd}. The objective function $J(\overline{\mathbf{x}}(\mathbf{P}_{\mathbf{k_{CP}}}), \hat{\mathbf{x}})$ describes the difference between these shapes as a scalar value and has to be minimized.

3.4. IMO Framework

The PYTHON-based injection molding optimization (IMO) framework implements the optimization problem explained in the section above. This section provides an overview of the optimization loop and describes the corresponding class structure of the implemented shape optimization tool. Additionally, a demonstration of a 2D optimization is given followed by a description of a sensitivity analysis method as well as a proposal for an initial guess.

The entire loop of the shape optimization for injection mold design is illustrated in Figure 7. The dashed boxes mark different parts of the algorithms using different libraries, which are depicted in Section 2.4.

Starting from an initial draft $\tilde{\Omega}$, the first numerical solution of the injection mold process will be determined according to the equations from Section 3.1. The results of the heat conduction, as well as, the shrinkage and warpage simulations are collectively referred to the result domain $\overline{\Omega}$.
The next step is the evaluation of the objective function J. With the *reference domain* $\hat{\Omega}$ a scalar value for the differences between the two shapes is determined according to the definition of Section 3.3.1. If the function value of J does not converge to the tolerance limit, the algorithm varies the position of the control points $\mathbf{P}_{\mathbf{k_{CP}}}$.
The new set of control points is passed to the free-form deformation and the deformed configuration Ω^{ffd} is returned. This shape is used as input for the next simulation. Once the algorithm determines the minimum of the objective function, the optimization loops stops and returns the *optimal design* for the cavity.

Following the description of the optimization loop for the presented framework, it is depicted which software tool from Section 2.4 is responsible for which component of the optimization. An overview of the software dependencies as well as the functionalities of the components is provided in Table 3

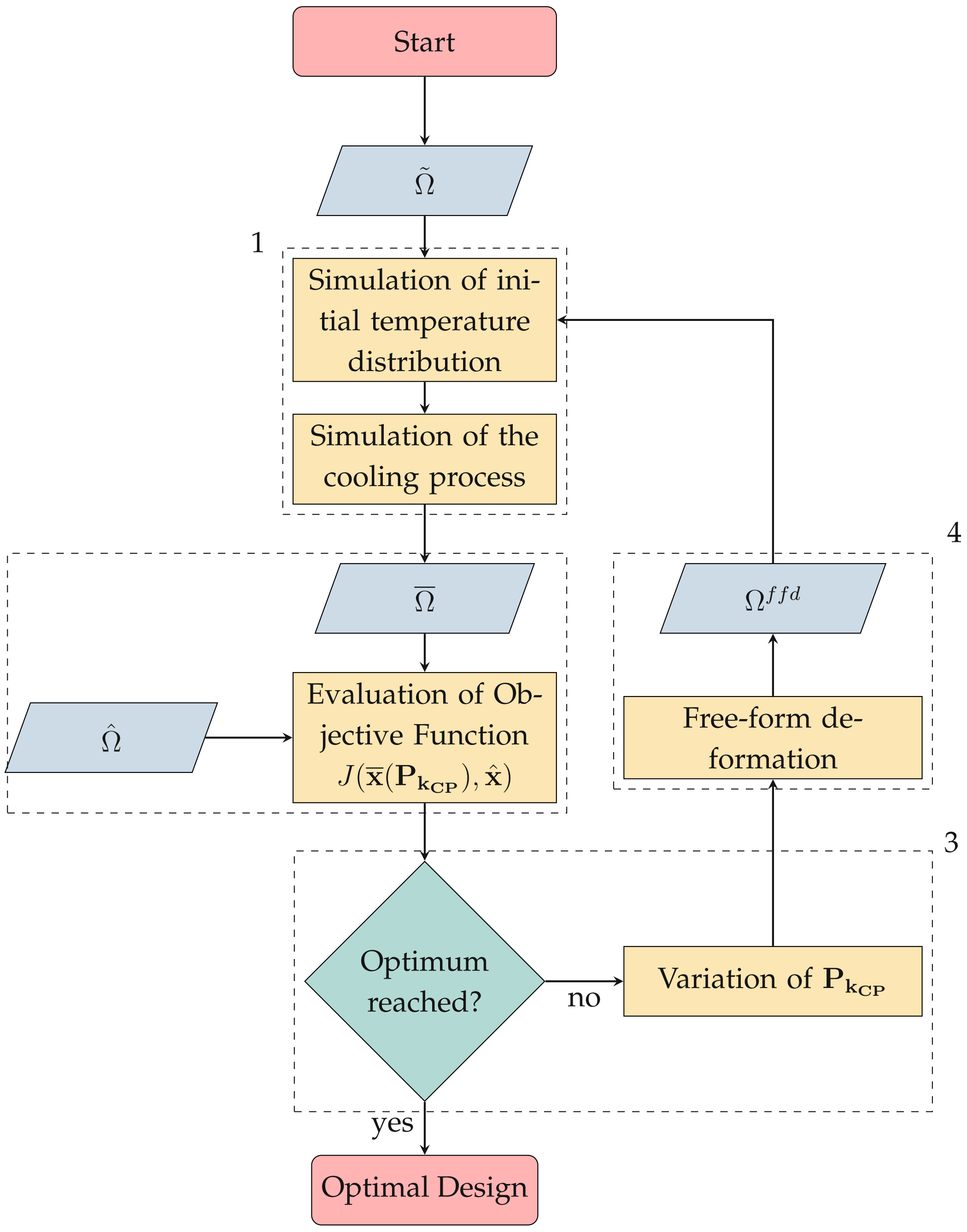

Start
$\tilde{\Omega}$
1
Simulation of initial temperature distribution
Simulation of the cooling process
2
$\overline{\Omega}$
$\hat{\Omega}$
Evaluation of Objective Function $J(\overline{\mathbf{x}}(\mathbf{P}_{\mathbf{k_{CP}}}), \hat{\mathbf{x}})$
4
Ω^{ffd}
Free-form deformation
3
Optimum reached?
no
Variation of $\mathbf{P}_{\mathbf{k_{CP}}}$
yes
Optimal Design

Figure 7: Optimization Loop. Dashed boxes mark the different components/programs of the framework with the indices referring according to Table 3. The initial geometry $\tilde{\Omega}$ is passed to the simulations (1). The result geometry $\overline{\Omega}$ is compared with the reference geometry $\hat{\Omega}$ by the objective function J (2). Until the optimum is reached, the algorithm (3) varies the control points $\mathbf{P}_{\mathbf{k_{CP}}}$. Based on the new set of control points, the updated geometry Ω^{ffd} from the free-form deformation (4) initializes the next simulation.

Table 3: Software dependencies for the IMO framework. The Indices refer to the optimization loop illustrated in Figure 7.

Index	Software	Description	Input	Output
1	NUTILS	Simulation of the temperature distribution and the cooling process.	$\tilde{\Omega}$ or Ω^{ffd}	$\overline{\Omega}$
2	NUTILS, PCU	Evaluation of the objective function $J(\overline{\mathbf{x}}, \hat{\mathbf{x}})$. The software is depending on the chosen objective function.	$\overline{\Omega}, \hat{\Omega}$	$J(\overline{\mathbf{x}}, \hat{\mathbf{x}})$
3	PDFO, SCIPY	The optimization of the objective function. The software is depending on the chosen optimization algorithm.	$J(\overline{\mathbf{x}}, \hat{\mathbf{x}})$	$\mathbf{P}_{\mathbf{kCP}}$
4	GUSTAF	The free-form-deformation of the initial shape.	$\mathbf{P}_{\mathbf{kCP}}$	Ω^{ffd}

3.4.1. Class Structure

Figure 8 shows the dependencies between the different classes of the IMO framework. In the following, these classes and their main functionalities will be discussed briefly.

Simulation

The framework is constructed in such a way, that the simulation class is passed to the framework externally. Therefore, the physical problem of interest can be defined freely. The displacement field $\mathbf{u}$ is passed from the shrinkage and warpage simulation.

nuguMesh

All meshes within the framework are a combination of two mesh types, namely the GUSTAF-mesh as well as the domain and geometry (geom) from the corresponding NUTILS-mesh. This is necessary in order to address both the FFD and the simulation from the same mesh object. With the current ControlPointsSet, a FFD of the initial mesh is performed.

Functional

The *Functional* class stores four different meshes, which are related to the different states of the configurations mentioned in Section 3.2. Further, this class points to the chosen objective function and stores the current objective function value $J(x) = D_J$.

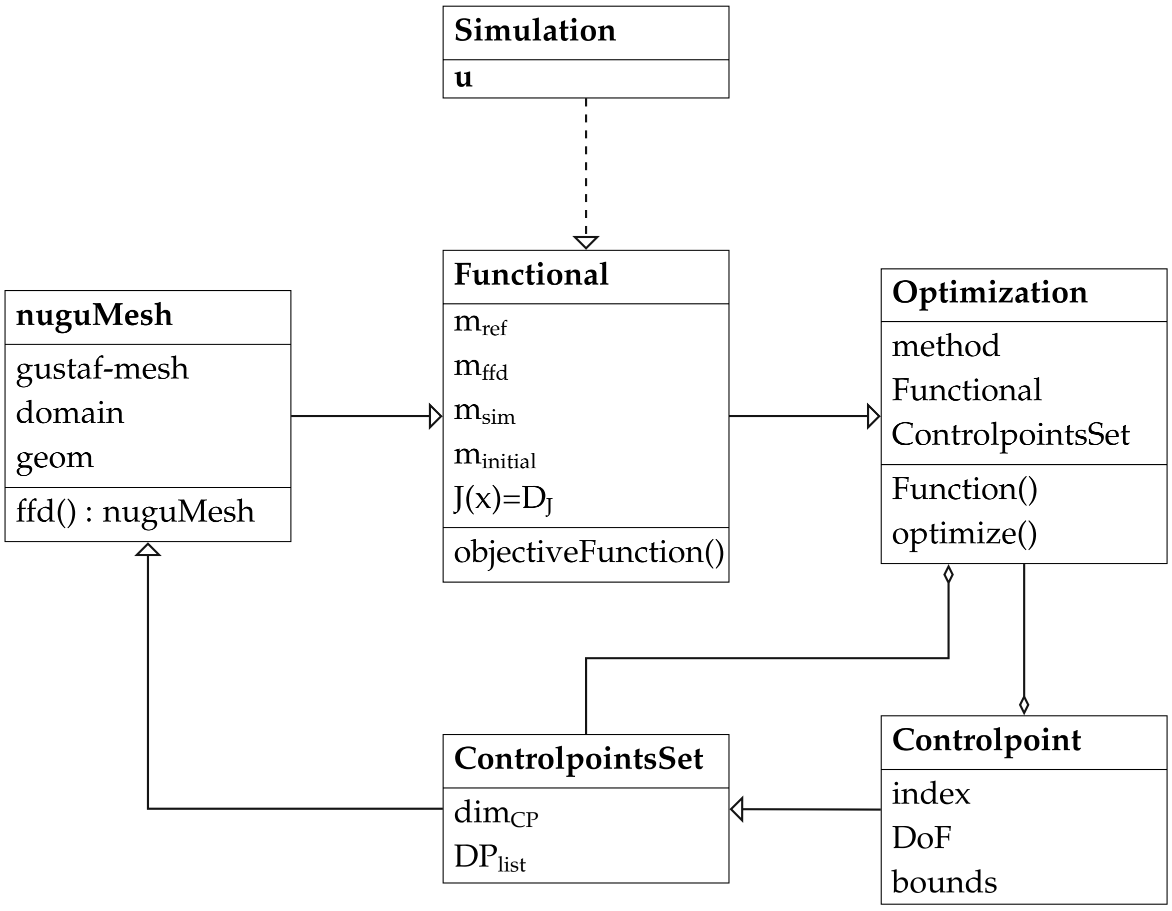


Figure 8: Class structure of the IMO framework. Arrows show strong relations, where the whole object is passed to another class. Lines marked with a diamond mark dependencies, but not as strong as the ones mentioned before. The dashed arrow shows an external dependency.

Controlpoint
The Controlpoint class stores all necessary information defining one control point $\mathbf{P}_{\mathbf{k}_{CP}}$ for the FFD spline. A control point is depicted by its index, the degree of freedom (DoF) and the bounds for the optimization.

ControlpointsSet
This class defines the dimensions dim_{CP} for the control point mesh of the FFD (see Section 3.3.2). The variable DP_{list} (DP = design parameter) defines, which control point $\mathbf{P}_{\mathbf{k}_{CP}}$ is allowed to be moved with respect to the specified degree of freedom.

Optimization
Finally, the *Optimization* class combines the *Functional* and *ControlpointsSet* classes. The preferred optimization algorithm optimizes the *dummy function* Function(), which calls the FFD and the simulation from the inside of the optimization loop. Different optimization algorithms can be called from here.

3.4.2. Optimization Representation in 2D

In this section, the introduced framework is represented as an example for a 2D mesh consisting of 4×4 elements, where each element has the size of 0.5×0.5 units. For the shrinkage and warpage simulation, the mesh is fixed at the center and no rotation is allowed (simulation parameters in Table 12). Further, we define a set of 4×4 control points with two degrees of freedom each,

which is why an optimization using BOBYQA using the *Distance Function* with 32 design parameters is carried out (see n=0). Goal of this optimization is to obtain a molding shape, which has the same shape as the initial mesh.

Each optimization loop in this framework starts with the simulation of the initial mesh $\tilde{\mathbf{m}}$ (n=0), where a significant discrepancy between the targeted shape $\hat{\mathbf{m}}$ and the reference shape $\tilde{\mathbf{m}}$ can be observed.

Initially, the algorithm begins with moving every design parameter. For n=1, the control point $\mathbf{P_0}$ is moved along its first degree of freedom, which, in this case corresponds to the x-axis. The deformed mesh $\mathbf{m}^{\mathbf{ffd}}$ is passed to the simulation, and the result mesh $\overline{\mathbf{m}}$ is compared with $\hat{\mathbf{m}}$.

The second function evaluation is performed for the deformation according to the second DoF of $\mathbf{P_0}$, which implies that $\mathbf{P_0}$ is moved along the y-axis. After this procedure is repeated for each design parameter, the algorithm determines a new approximation of the model to perform further function evaluations.

At function evaluation 150, a convexity can be observed at the bottom and the right side of $\mathbf{m}^{\mathbf{ffd}}$, but the simulation does not yield a satisfactory result, as the result mesh still deviates significantly from the desired shape. However, an improvement at the bottom can be observed.

Finally, after 5545 function evaluations, the algorithm returns the meshes as illustrated for n = 5545. The result shape closely matches the reference shape, and the objective function value is $D_J = 1.85310 \cdot 10^{-4}$, which is considered acceptable.

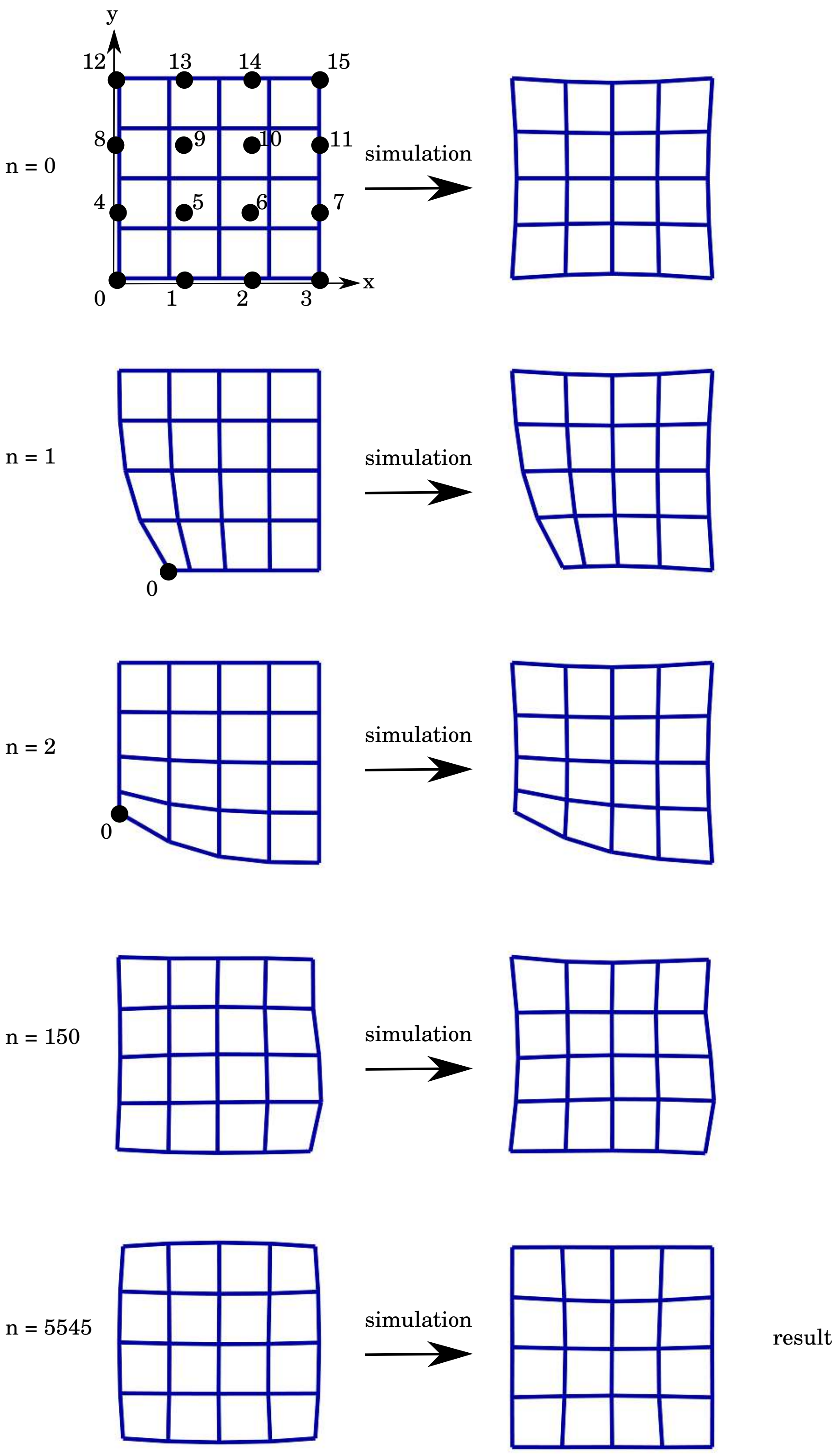


Figure 9: Representation of the optimization for a 2D square. On the left, the current initial mesh is shown, and the resulting mesh after the simulation on the right.

3.4.3. Sensitivity Analysis

The sensitivity S of the objective function J is defined as the partial derivative with respect to a design parameter. Since a design parameter corresponds to the spatial component $i = x, y, z$ of a single control point indexed k_{CP}, it can be expressed as

$$S_i^{k_{CP}} := \frac{\partial J}{\partial x_i^{k_{CP}}}. \tag{36}$$

Since we evaluate the sensitivity of a control point along one certain axis, we observe the sensitivity of a design parameter. Due to the lack of information of the derivative of the objective function the finite difference method is used as an approximation of the derivative [29]. Then, the sensitivity yields

$$\frac{\partial J}{\partial x_i^{k_{CP}}} \approx \frac{J|_{x_i^{k_{CP}}+h} + J|_{x_i^{k_{CP}}-h}}{2h}, \tag{37}$$

with h being the step size of the finite difference method. In this work, we assume $h = 10^{-6}$. By determining the sensitivities of the objective function, we can hierarchically order the influence of the individual design parameters, and therefore reduce the number of design parameters by a threshold. We define the threshold r as

$$\frac{S_i^{k_{CP}}}{S_i^{max}} < r, \tag{38}$$

where S_i^{max} is the maximum sensitivity in a certain spatial direction. By selecting an appropriate value for r, we can effectively eliminate design parameters that have minor influence on the optimization process.

3.4.4. Initial Guess

To speed up convergence of the optimization problem, an enhanced initial guess is needed. Since the shrinkage and warpage simulation returns a displacement field $\mathbf{u}$, an initial guess could involve utilizing the inverse of the displacement field.

Therefore, by initializing with the target domain $\hat{\Omega}$ as the input for the whole simulation of the process, the computed displacement field $\bar{\mathbf{u}}$ after the shrinkage and warpage simulation is defined as

$$\bar{\mathbf{u}}(\tilde{\mathbf{x}}), \tilde{\mathbf{x}} \in \tilde{\Omega}. \tag{39}$$

We define the initial guess by subtracting the displacement field from the initial coordinates. The *initial guess domain* then is

$$\Omega^{IG} := \{\mathbf{x^{IG}} \in \mathbb{R}^d : \mathbf{x^{IG}} = \tilde{\mathbf{x}} - \overline{\mathbf{u}}(\tilde{\mathbf{x}}), \tilde{\mathbf{x}} \in \tilde{\Omega}\}. \tag{40}$$

4. Evaluation and Analysis of Test Cases

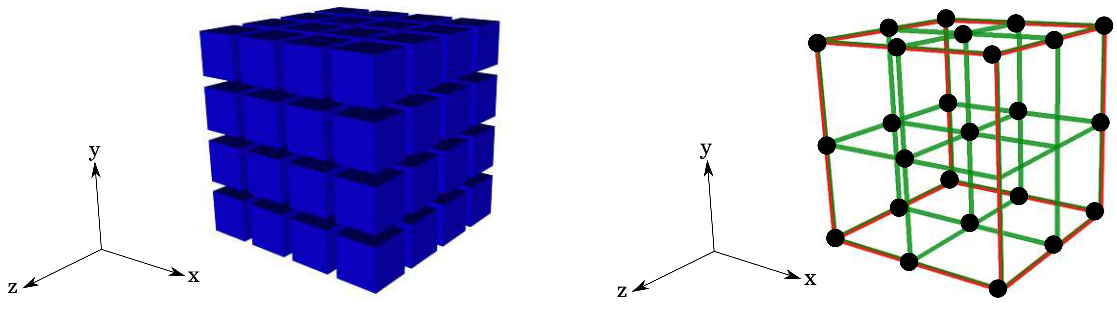
The presented framework is subjected to various tests. First, two test cases are defined: a cube and a L-Shape. Based on these two test cases, five different studies are carried out: (1) test with different algorithms, (2) influence of the choice of design parameters, (3) performance of a sensitivity analysis, (4) check of the initial guess and (5) test of the two objective functions.

4.1. Definition of Test Cases

In order to evaluate the proposed optimization scheme in the following chapters, two test cases are defined in this section. The first consists of a 3D cube, while the second test case has the 3D shape of an L. The cube yields a very basic testcase with high symmetries, whereas the L-Shape represents a geometry with internal corners and a varying wall thickness, in order to test the algorithms robustness.

4.1.1. Cube

The initial mesh of the cube with an edge length of 2 units consists of 64 hexahedral elements and is illustrated in Figure 10a. The initial mesh serves as the reference mesh, i.e. the aim of the optimization is to ensure that the mesh after the simulation has an identical shape as the initial mesh after the simulation. As shown in Table 4 we use 3 control points per dimension, implying that we obtain 27 control points in total. They are located symmetrical on the domain as seen in Figure 10b.



(a) Mesh of the cube (b) Control Points

Figure 10: (a) The elements are pulled apart for this visualization of the cube test case. (b) The control points are symmetrically distributed over the domain. The outline (red) marks the spline of the undeformed mesh. Black points mark the location of the control points.

Table 4: Cube test case parameters.

Dimension	$2 \times 2 \times 2$
Nodes	$5 \times 5 \times 5$
Control points	$3 \times 3 \times 3$

The heat conduction and the cooling process are modelled as formulated in Section 3.1. The boundary temperature of the solid body is defined as

$$T_{Dirichelet} = 1 \text{ on } \Gamma, \tag{41}$$

where Γ describes the surface of the solid body. The initial temperature is set to

$$T_{Initial} = 100. \tag{42}$$

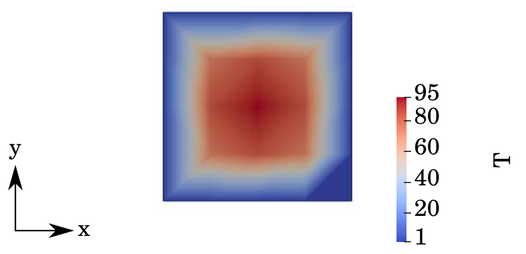
Also, the rigid body movement is restricted by setting the displacement at the center of the cube to

$$u_{x=1,y=1,z=1} \overset{!}{=} 0. \tag{43}$$

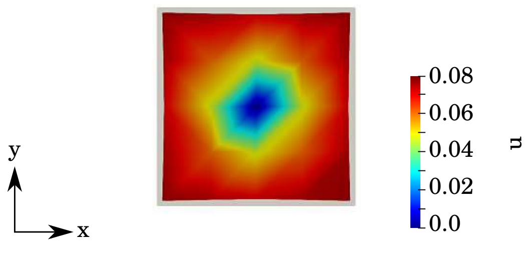
Additionally, in order to make the problem statically determinant, the rotation around all axes is restricted. All relevant parameters for the simulation are listed in Table 13.

The temperature distribution obtained by the heat conduction simulation (time = 0.2 seconds) is depicted in Figure 11a. The maximum of the temperature is located in the center of the cube, while the boundary temperature on the surfaces is constant. We denote, that the temperature displaying in some corners is not consistent, even though the determined values are correct. This could be caused by the support of triangulation of graphical processors.

Figure 11b shows the cross-section of the deformed mesh, with the color scale indicating the magnitude of displacement caused by shrinkage. In the background (in light gray), one can see the initial mesh before the simulation. The deformed mesh has shrunk in comparison with the initial mesh. Due to the boundary conditions, the displacements are smallest in the center, whereas the largest displacements occur at the corners.



(a) Temperature distribution



(b) Displacement field

Figure 11: (a) Temperature distribution after 0.2 seconds for the inner part of the cube. The maximum temperature is in the center. (b) Displacement field for the cross-section of the cube. The initial form is illustrated behind in gray. Since no real material properties are modelled, the utilization of units becomes unnecessary.

4.1.2. L-Shape

The L-Shape test case consists of 320 elements, with a cross-section of 4×4 elements and a side length of 12 elements as illustrated in Figure 8. This test case is chosen in order to find out, if the presented method is suitable to handle corners of the geometry. Since corners have a change in the cross-sectional area, warpage typically occurs in these regions. Additionally, this shape does not show as many symmetries as the cube. The rigid body movement is not allowed, which means

$$u_{x=0,y=0,z=0} \overset{!}{=} 0, \tag{44}$$

and as explained above, any rotation around all axes is restricted.

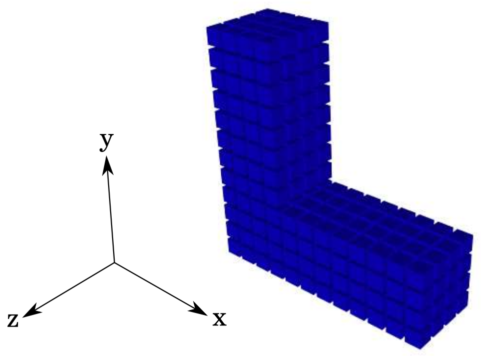
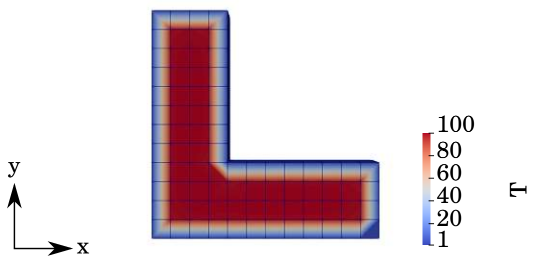


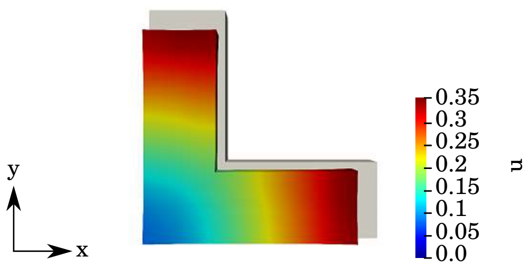
Figure 12: Geometry of the L-Shape test case. The elements are pulled apart for this visualization.

The temperature distribution after the heat conduction simulation is calculated with the same initial temperature values as in the above test case and is illustrated in Figure 13a. For the relevant simulation parameters, refer to Table 14.

The displacements from the shrinkage and warpage simulation are shown in Figure 13b, scaled by a factor 5, which allows a better observation of warpage effects. Notably, curvature can be seen the outer surfaces (x_{max} and y_{max}).



(a) Temperature distribution



(b) Displacement field

Figure 13: (a) Temperature distribution after 0.2 seconds for the inner part of the L-Shape. (b) Displacement field (scaled by factor 5) for the L-Shape. The initial form is illustrated behind in gray. Since no real material properties are modelled, the utilization of units becomes unnecessary.

4.2. Algorithm Test

In order to assess the compatability and performance within the IMO framework, the four algorithms presented in Section 2.3.2 will now be examined. For this purpose, the two introduced test cases, namely the cube and L-Shape, will be evaluated.

4.2.1. Cube Test Case

First, we define the reference mesh as described in Table 4 which also serves as the initial mesh of the optimization. Therefore, no enhanced guess about the optimization solution is passed to the algorithm. The objective function is described through the *Distance Function* as described in Section 3.3.1 and the solver's specific tolerance level is set to 10^{-6}

The control mesh is defined by $3 \times 3 \times 3$ grid of control points and since movement in all three spatial directions is allowed, this yields in a set of 81 design parameters (see Table 5). Each design parameter is constrained with the bound of (-0.5, 0.5), limiting the movement of each design parameter to $\pm$ 0.5 units. Since BFGS cannot handle any constraints, these bounds can be disregarded for all optimizations with BFGS.

Table 5: Cube test case design parameters.

Index	DoF x	DoF y	DoF z
0-26	(-0.5, 0.5)	(-0.5, 0.5)	(-0.5, 0.5)

The result of this test case is presented in Table 6. Among the four algorithms, BFGS shows fastest convergence, requiring 2214 function evaluations. BOBYQA algorithms terminates after 4610 function evaluations, while COBYLA and LINCOA are stopped after 40500 function evaluations due to default settings of the solver, as they failed to converge. The last column of the table indicates the number of required function evaluations in order to reach a value below $< 6 \cdot 10^{-3}$. It should be noted, that LINCOA reaches this value needing only 200 function evaluations more than BOBYQA.

Since all algorithms converge to nearly the same value, and the value of the objective function is sufficiently small, it can be observed that the optimum is reached for this specific test case. As the number of design parameters is limited, further improvement in the results can not be achieved.

Figure 14 illustrates the objective function values in a double-logarithmic scaling. The fastest convergence is achieved by BFGS algorithm (in red). Additionally, BOBYQA (in blue) and LINCOA (in orange) reach a low value at around 2000 function evaluations, whereby the slower convergence rate of LINCOA can be observed. The figure clearly demonstrates that COBYLA (in green) algorithm is remarkable slower at converging for this test case.

Table 6: Optimization results for the cube test case with 81 design parameters. The variable n denotes function evaluations needed in order to reach the minimum of the optimization.

Algorithm	n	Minimum	$< 6.3 \cdot 10^{-3}$
BFGS	2214	$6.29589 \cdot 10^{-3}$	1067
BOBYQA	4610	$6.29591 \cdot 10^{-3}$	1956
COBYLA	40500	$6.29642 \cdot 10^{-3}$	8379
LINCOA	40500	$6.29590 \cdot 10^{-3}$	2159

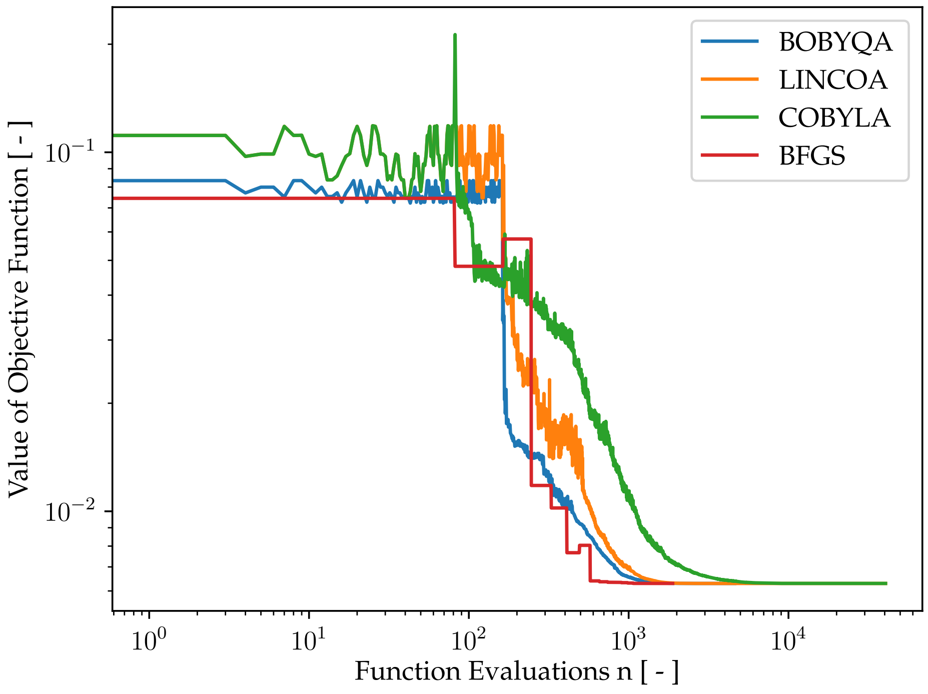


Figure 14: Plot of the *Distance Function* for the cube test case with 81 design parameters.

Results BFGS Optimization

As all optimizations converge to similar values, it is sufficient to only show one final output. Therefore, the results from the BFGS optimization are depicted in this section.

In Figure 15 we can see the cross section (normal to z-axis) of the deformed mesh $\mathbf{m}^{\text{ffd}}$ of the cube and the cross section (normal to x-axis) of the initial mesh $\tilde{\mathbf{m}}$ in gray. The coloring indicates the magnitude of the deformation due to FFD. As defined in the boundary conditions, the center of the cube is permitted to undergo any movement. Therefore, the deformation arises from the inside to the surface of the cube, and highest values are obtained for the corners.

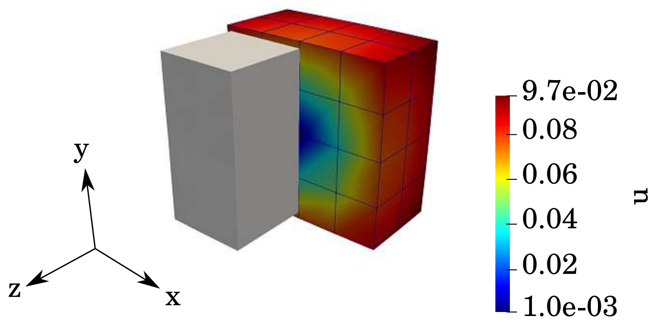


Figure 15: FFD of the cube after the optimization. The gray mesh shows half of the initial mesh. Since no real material properties are modelled, the utilization of units becomes unnecessary.

The difference between the mesh after the simulation $\overline{\mathbf{m}}$ and the reference mesh $\hat{\mathbf{m}}$ is illustrated in Figure 16. The greatest difference at the surface is $d = 0.011$, which corresponds to only 0.55% of the edge length.

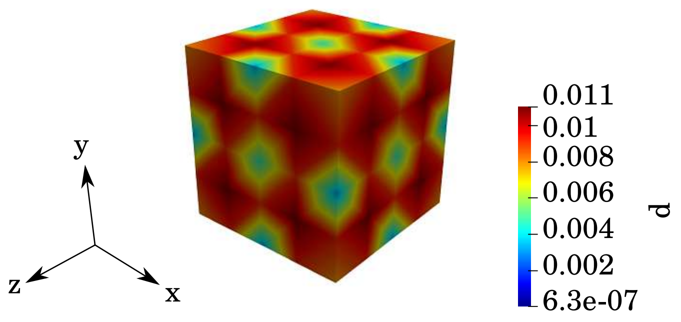


Figure 16: Difference d between the simulated mesh and the reference mesh. For a perfectly optimized molding shape, all discrepancies would be zero. Since no real material properties are modelled, the utilization of units becomes unnecessary.

The least differences occur at the middle of an edge or surface, which could be caused by the presence of control points at exactly these locations. However, this does not apply for the corners of the cube, where the difference between $\overline{\mathbf{m}}$ and $\hat{\mathbf{m}}$ is noticeable bigger. The maximum differences between both meshes occur at the inner nodes of a surface.

4.2.2. L-Shape Test Case

The performance of the different algorithms investigated above is now examined for the L-Shape (see Section 4.1.2) with a net of $3\times3\times3$ control points as shown in Figure 17. For the control point bounds, the same values are used as in the previous test case (see Table 5), the specific parameters for the simulation are listed in Table 14 and the solver's tolerance limits are set to 10^{-6}.

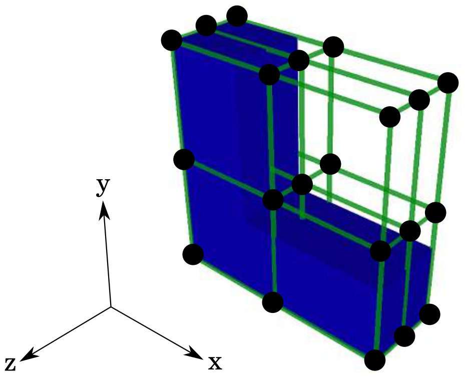


Figure 17: Control point mesh with 27 CP for the L-Shape test case.

The final result of the optimizations is illustrated in Table 7. In this specific test case, it is observed that only the algorithms utilizing quadratic approximations of the model (BFGS, BOBYQA and LINCOA) converge successfully. In contrast, COBYLA was interrupted automatically, as it did not converge within a suitable timeframe. Similar to the cube test case, BFGS demonstrates the fastest convergence, while BOBYQA and LINCOA also show good convergence behavior.

Table 7: Optimization results of the L-Shape test case with 81 design parameters. The variable n describes the function evaluations needed in order to reach the minimum of the optimization.

Algorithm	n	Minimum	$< 3.8 \cdot 10^{-3}$
BFGS	10168	$3.79204 \cdot 10^{-3}$	5577
BOBYQA	31977	$3.79204 \cdot 10^{-3}$	7493
COBYLA	40500	$4.33890 \cdot 10^{-3}$	-
LINCOA	33559	$3.79204 \cdot 10^{-3}$	10449

Figure 18 illustrates the values of the objective function over the function evaluations in a double logarithmic scaling. In comparison with the cube test case, BOBYQA (in blue) as well as LINCOA (in orange) show an improved convergence behavior. However, the BFGS (in red) algorithm outperforms both of them at the end of the optimization.

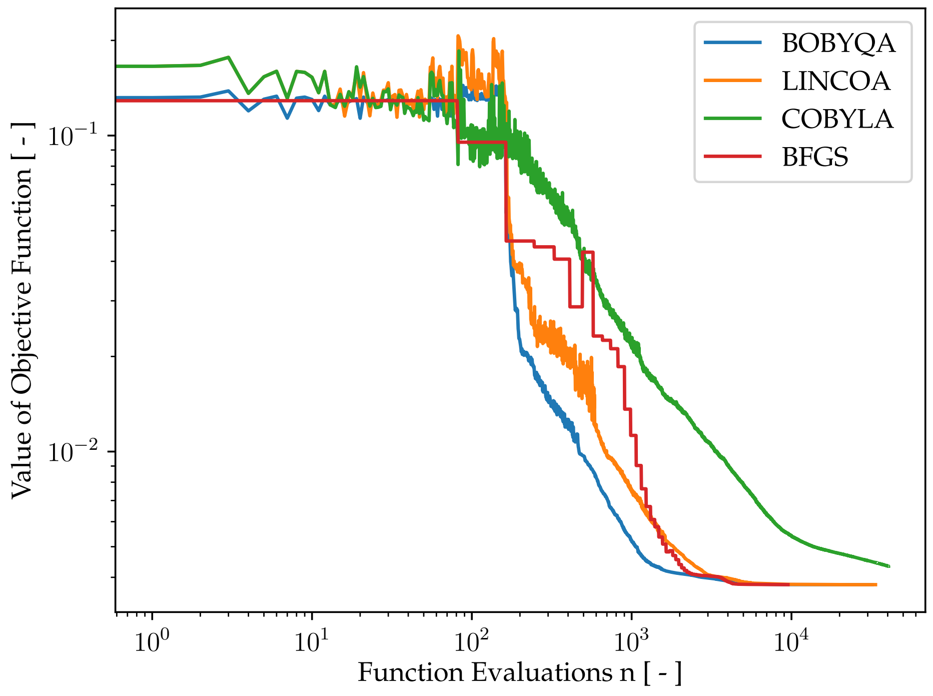


Figure 18: Plot of the *Distance Function* for the L-Shape test case with 81 design parameters.

4.3. Design Parameter Test

In order to prove the influence of the choice and number of design parameters, both of the test cases from the previous section will be performed again, but with a different set of design parameters.

4.3.1. Cube Test Case

The same test case from Section 4.2.1 is used with a reduced set of 54 design parameters. The reduced set of design parameters is chosen, such that such that unnecessary directions in the control point movement are blocked. E.g. it is not possible to move a control point in direction to the center of the cube, as we know that the cooling of the solid will cause shrinking, and therefore, the cube has to expand in order to reach the optimal shape. Additionally, all symmetries are taken into account and a movement of the control point located at the center of the cube is not needed. These conditions result in the set of design parameters documented in Table 16. Parameters for the simulation are listed in Table 13 and the tolerance level is 10^{-6}.

The results from this test are shown in Table 8. Again, BFGS shows the best performance, while BOBYQA and LINCOA have similar numbers of iterations. In comparison to the results obtained from the 81-design parameter test case (see Figure 14) with unrestricted bounds, all show a significantly improved performance. The BFGS algorithm needs approximately 72% of the function evaluations, while BOBYQA needs around 56%. LINCOA needs 20.3% of the function evaluations, and the most significant decrease is measured for COBYLA with 5.6% of the previous function evaluations.

Table 8: Optimization results of the L-Shape test case with 81 design parameters. The variable n describes the function evaluations needed in order to reach the minimum of the optimization.

Algorithm	n	Minimum
BFGS	1595	$6.29589 \cdot 10^{-3}$
BOBYQA	2586	$6.29590 \cdot 10^{-3}$
COBYLA	2669	$6.29590 \cdot 10^{-3}$
LINCOA	8223	$6.29590 \cdot 10^{-3}$

The objective function values for this test case are illustrated in Figure 19. A rapid decline can be observed for BOBYQA (in blue), whilst LINCOA (in orange) and COBYLA (in green) perform comparatively worse. Towards the end of the optimization, BFGS (in red) outperforms the other algorithms. While the trends for BFGS, BOBYQA and LINCOA are comparable to the previous testcases, COBYLA shows improved convergence behavior.

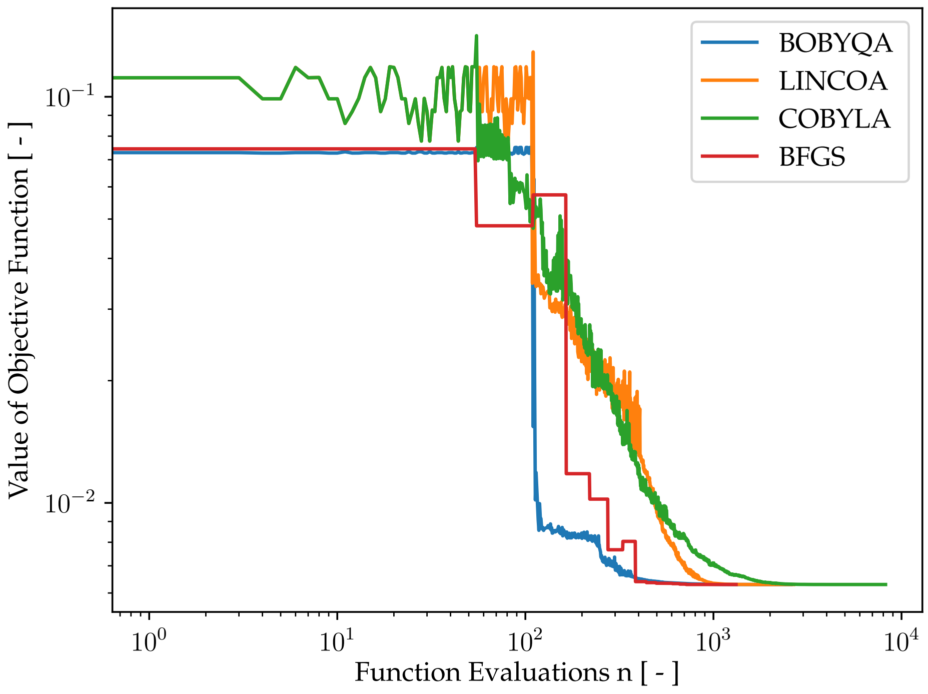


Figure 19: Plot of the *Distance Function* for the cube test case with 54 design parameters.

4.3.2. L-Shape Test Case

The aim of the study is to identify the relationship between the function evaluations needed and the determined minimum of the objective function for an increasing number of design parameters. Therefore, nine different sets of control meshes with an increasing number of control points are defined as shown in Table 9. Simulation parameters are listed in Table 14 and the tolerance level is set to 10^{-5}, so we expect a faster convergence of the optimization.

Table 9: L-Shape test case design parameters.

Dimension	Number of CP	Number of DP
$2 \times 2 \times 2$	8	24
$3 \times 3 \times 2$	18	54
$3 \times 3 \times 3$	27	81
$4 \times 4 \times 3$	48	144
$4 \times 4 \times 4$	64	192
$5 \times 5 \times 3$	75	225
$7 \times 7 \times 3$	147	441
$8 \times 8 \times 4$	256	768
$10 \times 10 \times 4$	400	1200

Figure 20 depicts the final values of the objective function for each set of design parameters. For the tests with 441, 768 and 1200 design parameters, the optimization was stopped manually at 200.000 function evaluations, as they did not reach the tolerance limit of the algorithm. The curve shows a hyperbolic behavior; an increasing number of design parameter leads to a decrease of the objective function value. When considering a lower number of design parameters, a significant drop of the objective function can be observed. In contrast, as the number of design parameters increases, the final value decreases at a slower rate. Since the number of needed function evaluations increases with a higher number of design parameters (compare Section 4.3.1), the computational time rises dramatically. Optimizing with a set of $4 \times 4 \times 4$ control points $(2.23927 \cdot 10^{-3})$ results in a better outcome compared to the optimization with $5 \times 5 \times 3$ control points $(2.42506 \cdot 10^{-3})$, despite the number of design parameters is higher for the second case.

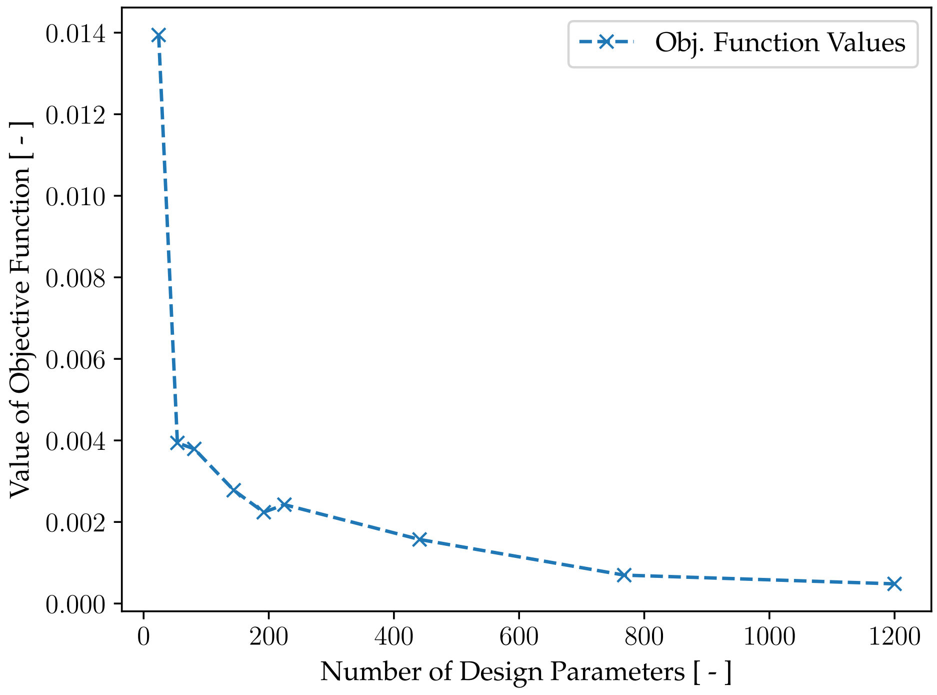


Figure 20: Objecive function values for different numbers of design parameters for the L-Shape test case.

4.4. Sensitivity Analysis

With a sensitivity analysis, the sensitivity of a single design parameter with respect to the objective function can be measured. The proposed method from Section 3.4.3 is tested in this section.

4.4.1. Cube Test Case

The sensitivity analysis is carried out for the cube test case with 81 design parameters (see Section 4.1.1). Table 13 shows the simulation parameters, the tolerance level for the BFGS algorithm is 10^{-5}.

With the step size $h = 10^{-6}$ (see Equation 37) and by using the three spatial components of a control point, we can describe the sensitivity for one control point by a vector. Then, the length (or norm) of this vector describes the magnitude of the sensitivity of the control point with respect to the objective function and the geometry.

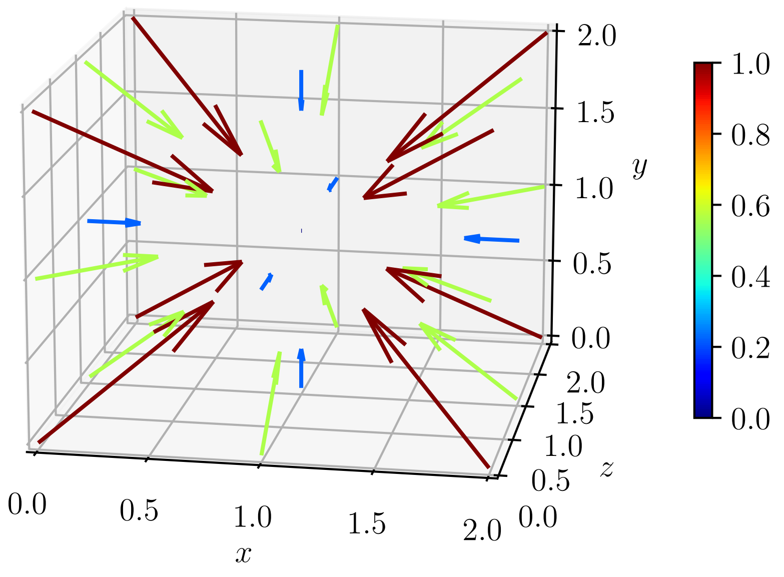


Figure 21: Field plot for the design parameter sensitivity of the cube. The arrows show the direction of the sensitivity; length and color indicate the value (norm) of the sensitivity. All vectors are normalized with a *min-max-scale*.

The determined sensitivities are illustrated as vectors, normalized with a *min-max-scale*, which means that the maximum values are normalized to one, and the minimum values are normalized to zero. Figure 21 shows the resulting vector field for the sensitivities of the cube's control points. The corners of the cube exhibit the highest sensitivity, whereas the sensitivity at the center is zero. This observation also underscores the symmetry of the problem.

With the threshold

$$\frac{S_i^{k_{CP}}}{S_i^{max}} < r, \tag{45}$$

we can eliminate design parameters with low sensitivities. By choosing $r = 0.05$, the new set of design parameters that was automatically selected this way coincides with the manually chosen set that was described in Section 4.3.1.

Optimization with the reduced set of design parameters show satisfactory results, as illustrated in Table 10. Due to symmetrical reasons, the objective function converges to the exact same value, while requiring approximately 70 % of the computational effort compared to the test with 81 control points.

Table 10: Optimization results of the cube test case with 81 design parameters for different values of r. The variable n describes the function evaluations needed in order to reach the minimum of the optimization.

r	Number of DP	n	Minimum
0	81	1886	$6.29590 \cdot 10^{-3}$
0.05	54	1320 (69.99 %)	$6.29590 \cdot 10^{-3}$ (100 %)

4.4.2. L-Shape Test Case

The sensitivity analysis is examined for the L-Shape by using a set $4 \times 4 \times 3$ control points, which yields all together 144 design parameters. As illustrated in Figure 22, control points which are not directly in contact with the geometry, are marked in gray.

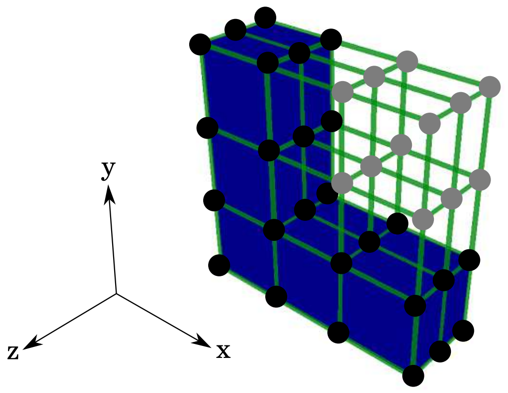


Figure 22: Control point mesh for the L-Shape test case with 48 control points. Gray marked control points are not in contact with the geometry.

We perform a sensitivity analysis (see Section 3.4.3) utilizing the BFGS-algorithm for all 144 design variables with a tolerance of 10^{-5}. The resulting normalized vector field for the control points sensitivites is shown in Figure 23.

The highest sensitivities are observed at the outer layer of the L-shape, specifically at $x = 0$ and $y = 0$, whereby the inner control points ($z = 1$) exhibit smaller sensitivities. In contrast, the corners $((x, y, z) = \{(0, 6, 0), (0, 6, 2), (0, 2, 0), (6, 0, 0), (6, 0, 2)\})$ mark control points with a high sensitivity to the problem.
The smallest values occur exactly in the corner (at $x = 4, 6$ and $y = 4, 6$), where the control points are not in contact with the geometry.

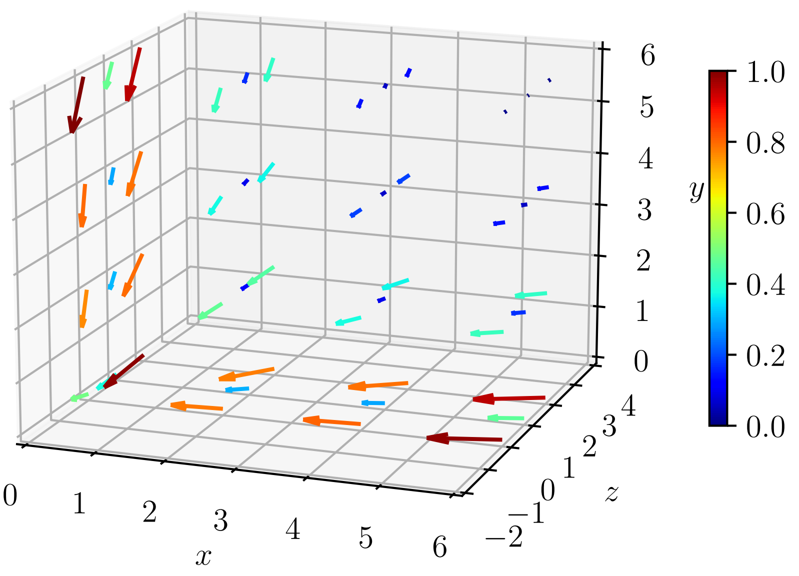


Figure 23: Field plot for the design parameter sensitivity of the L-Shape. The arrows show the direction of the sensitivity; length and color indicate the value (norm) of the sensitivity. All vectors are normalized with a *min-max-scale*.

In order to test if the number of design parameters can be reduced depending on the results obtained by the sensitivity analysis, two tests are performed. The threshold for the automatized elimination of insignificant design parameters is set to $r = 0.025$ or $r = 0.05$, which automatically generates two sets of 122 and 105 design parameters, respectively. These created sets (see Table 11) are tested against the original set (see Section 4.3.2).

In both optimizations with an automatically reduced set of design parameters, the objective function yields higher values compared to the reference test case. For $r = 0.05$, the final result is approximately 70% higher than the reference value, yet it requires less function evaluations (35.83%). Objective functions values for $r = 0.025$ are approximately 10% greater than those for $r = 0$ and the number of function evaluations are reduced by 46.94%.

Table 11: Results of the L-Shape test case optimization with 144 design parameters for different values of r. The variable n describes the function evaluations needed in order to reach the minimum of the optimization.

r	Number of DP	n	Minimum
0	144	36685	$2.77900 \cdot 10^{-3}$
0.025	122	17220 (46.94%)	$3.05544 \cdot 10^{-3}$ (109.95%)
0.05	105	13144 (35.83%)	$4.79849 \cdot 10^{-3}$ (169.43%)

The values of the objective function for the performed test cases are shown in Figure 24 in a double logarithmic scaling. The blue curve shows the evolution of the objective function for the reference test case with $r = 0$, the values for the automatically generated design parameter sets for $r = 0.025$ and $r = 0.05$ are illustrated in green and red, respectively. The red curve clearly shows the fast convergence, but with a significantly higher value compared to the other curves. The lines corresponding to $r = 0$ and $r = 0.025$ show a similar trend, with the line for $r = 0$ showing a drop towards the end of the optimization.

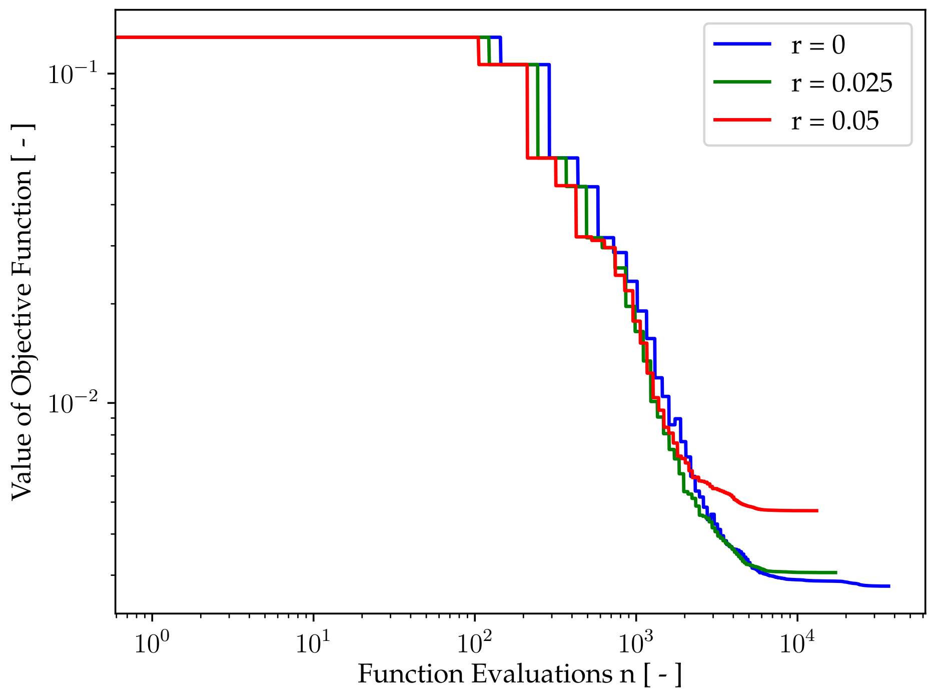


Figure 24: Plot of the objective function for the L-Shape test case for different values of r. With r, a new set of design parameters is automatically generated from a sensitivity analysis.

4.5. Initial Guess

With a suitable initial guess, the number of iterations for the optimization can be decreased. As defined in Section 3.4.4, the inverse displacement is added on the reference mesh and taken as initial mesh for the optimization. The effectiveness of this approach is tested in the following section for the cube and the L-Shape.

4.5.1. Cube Test case

Here, six different optimization tests are carried out in order to quantify the influence of the proposed initial guess. Four tests are performed with the cube shape as described in Section 4.1.1 using BFGS or BOBYQA. The 54 design parameters test (see Table 16) is also examined with the initial guess.

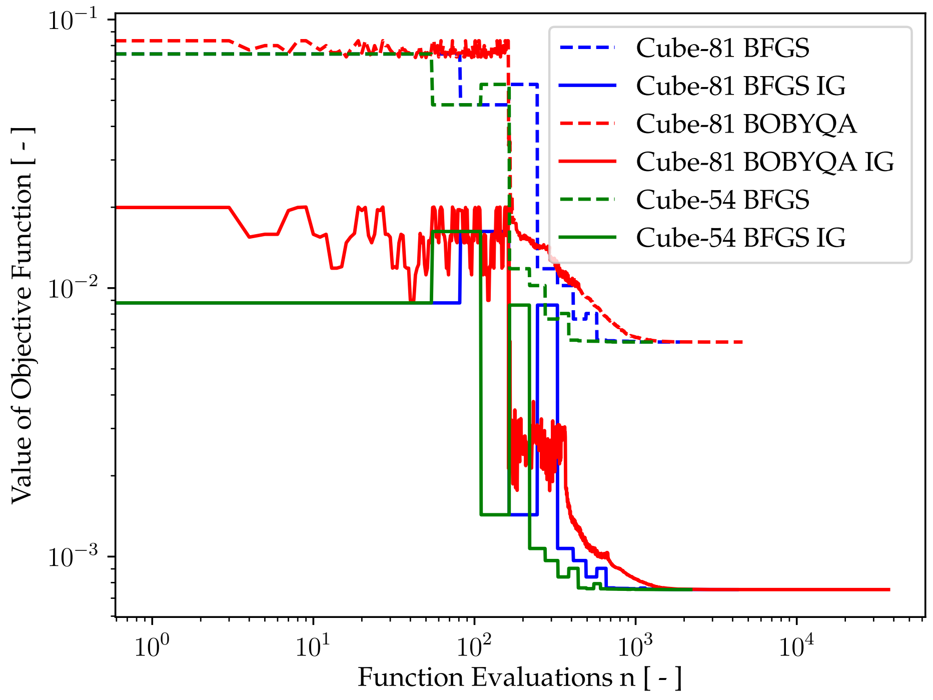


Figure 25: Cube tests with and without initial guess. The index IG denotes for which test the initial guess was used.

The results for all tests are illustrated in Figure 25 in a double logarithmic scale. Dashed lines mark the *original* results from the previously optimized test cases without using the initial guess. On the contrary, solid lines in the same color mark the corresponding results with the utilization of the initial guess as input mesh (suffix IG in the legend). At the beginning of the optimization, the tests with the initial guess show a significantly smaller value of the objective function. It should be emphasized that the objective function for both BFGS IG tests shows an intermediate increase.

All tests show a similar behavior, as the solid lines converge to a lower value of the objective function, which implies a more accurate result. Additionally, the optimization using the initial guess needs less function evaluations to converge. Final values for the objective function are noticeable smaller than these from the *original* test cases. For the Cube-81 IG, the final results are $8.106\ \%$ of the Cube-81 optimization with the BFGS algorithm and $11.969\ \%$ for BOBYQA. The Cube-54 BFGS IG test case's final value of objective function is $10.970\ \%$ of the Cube-54 BFGS.

4.5.2. L-Shape Test Case

The L-shape test case is carried out with 81 design parameters as previously shown in Figure 17) and 144 design parameters (see Figure 22), both using BFGS for the optimization routine.

Both optimizations using the initial guess (solid lines, suffix IG) show a faster converging behavior as illustrated in Figure 26. Both optimizations with the initial guess show a similar trend and result in a significant smaller value. While the L-81 IG result is only $6.108\ \%$ of the L-81 test, the L-54 IG test, with just $5.749\ \%$ of the L-54 test case's final value. At the beginning of the procedure, the objective function shows a peak for both initial guess tests.

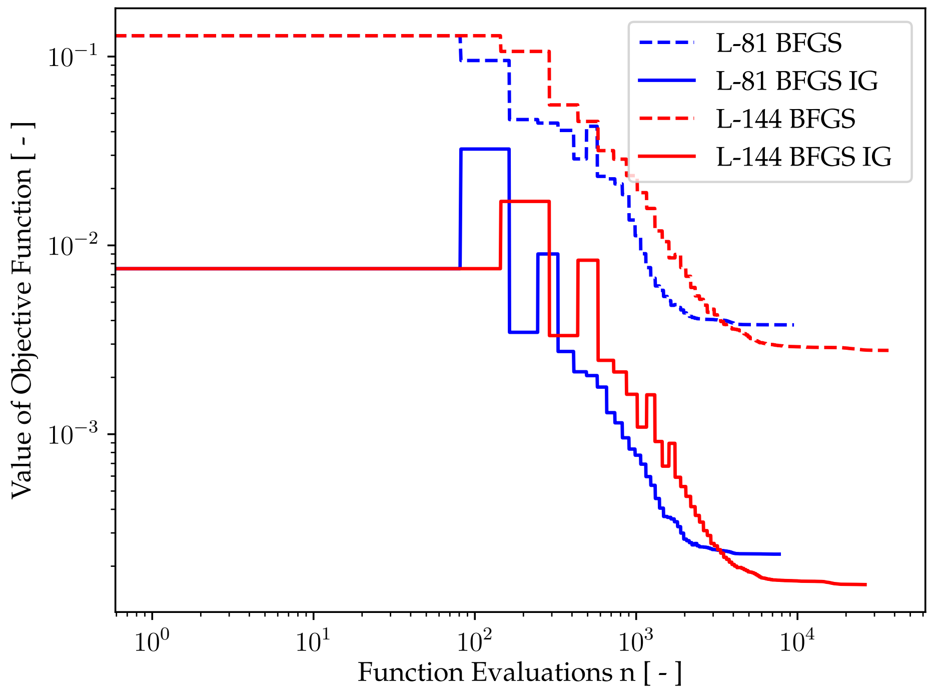


Figure 26: L-Shape tests with and without initial guess. The index IG denotes for which test the initial guess was used.

4.6. Objective Function Test

With this test, two different objective functions are being compared. So far, only the *Distance Function* (see Section 3.3.1) has been utilized. However, in this section, the performance of the *Hausdorff Distance* (see Section 3.3.1) is examined.

4.6.1. Cube test case

The cube test cases (refer to Section 4.1.1) with 81 design parameters (see Table 5) and 54 design parameters (see Table 16) are tested with the *Hausdorff Distance* as objective function instead of the *Distance Function*.
Since BFGS is a gradient-based optimization algorithm and the *Hausdorff Distance* is a piecewise linear function, we assume that a combination is not feasible or would not show acceptable results. Therefore, BFGS is neglected in this testing.
For the 61 design parameter test case, the results are shown in Figure 27. Only the BOBYQA and LINCOA algorithms show feasible results at the end of the optimization. Again, LINOCA, converges relatively slow (33525 function evaluations).

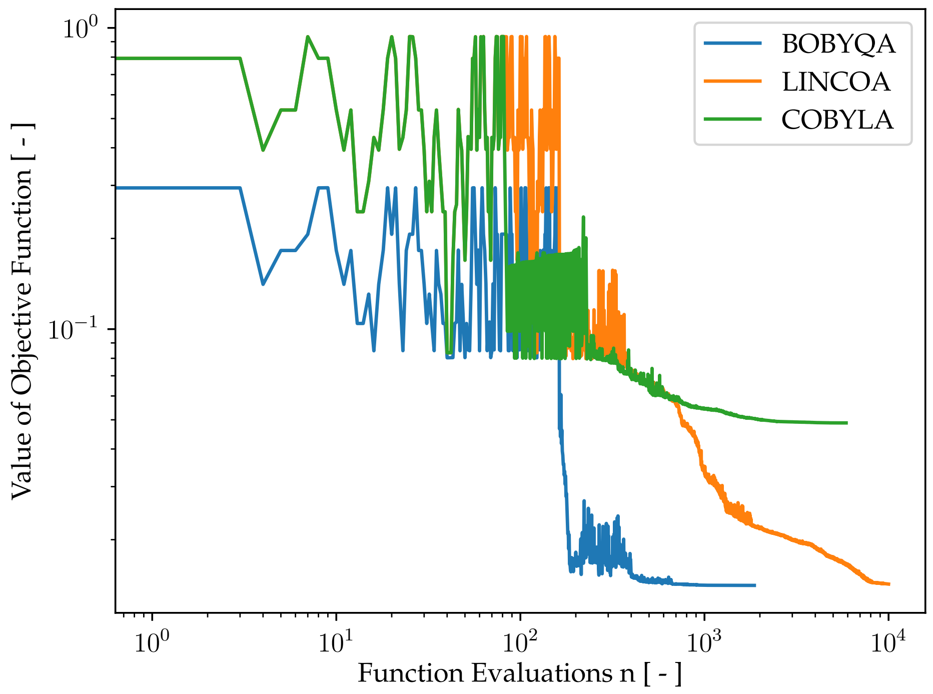


Figure 27: Objective function values for the cube test case with the *Hausdorff Distance*.

Figure 28 shows the comparison between the optimization with the *Hausdorff Distance* and *Distance Function* for the BOBYQA algorithm using a double logarithmic scale. In order to see the impact of a different objective function, we need to measure the second objective function as a reference, while optimizing the first objective function.

The solid blue line depicts the evaluated *Hausdorff Distance* for the optimization using the *Hausdorff Distance* as objective Function. The dashed blue line shows the corresponding *Distance Function* for the *Hausdorff* optimization. Red lines show the objective function values for the *Distance Function* optimization. The dashed red line marks the values for the *Distance Function*, while the solid red line shows the corresponding *Hausdorff Distance*.

The optimization with the *Hausdorff Distance* (in blue) converges at fewer function evaluations compared to the optimization with the *Distance Function* (in red). Looking at the measured *Distance Function* (dashed lines), the *Distance Function* performs much better. In contrast, both measured *Hausdorff Distances* converge to similar values.

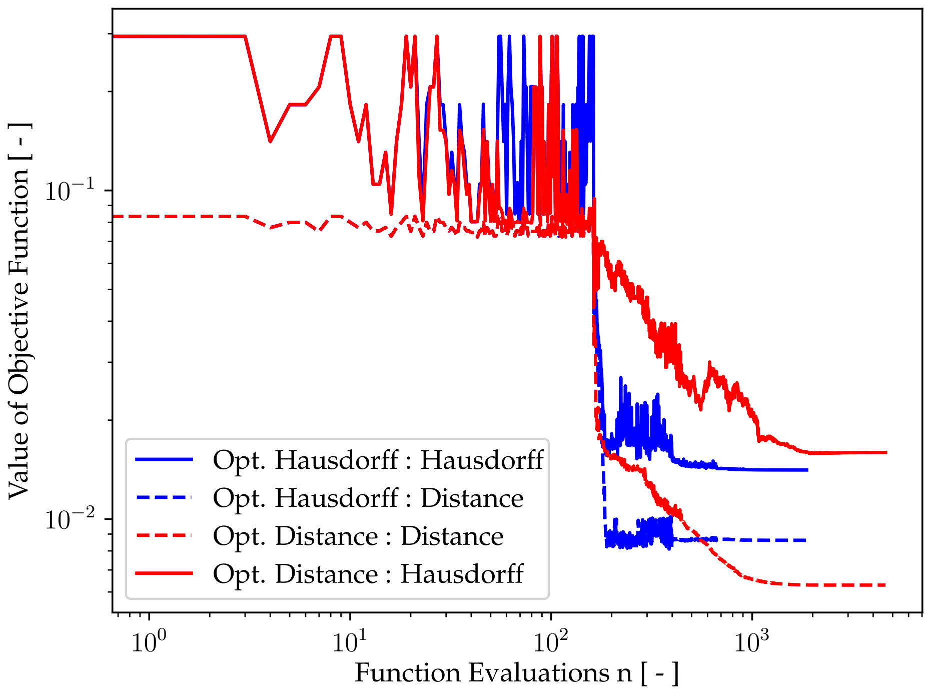


Figure 28: Comparison for the objective function values for a *Hausdorff Distance* and *Distance Function* driven optimization with BOBYQA. Opt. Hausdorff : Haussdorf denotes the optimization which minimizes the *Hausdorff Distance* and measures the *Hausdorff Distance*. Opt. Hausdorff : Distance denotes the optimization which minimizes the *Hausdorff Distance* and measures the *Distance Function* as a reference.

5. Discussion

Five tests have been carried out to investigate the performance of the IMO framework under different conditions: (1) testing different algorithms, (2) a design parameter study, (3) evaluating the automatized sensitivity analysis, (4) assessing the impact of the initial guess and (5) comparing two different objective functions. This section aims at discussing the results obtained from these test cases, along with the limitations of the framework.

5.1. Algorithms

An important component of any optimization routine is the choice of an appropriate algorithm. Therefore, four different algorithms were tested with regard to their compatibility and performance in combination with the presented framework (see Section 4.2).

The noise of the objective function during the optimization with COBYLA, BOBYQA and LINCOA are due to the larger range of tested values for the individual design parameters compared to BFGS. This implies that a a poorly chosen design parameter leads to a peak in the objective function.

The test cases investigating different algorithms for the cube and the L-Shape show that algorithms utilizing quadratic model approaches generally perform better than those utilizing linear approximations. As shown in Table 6 and 7, the BFGS algorithm shows the best result for both test cases. The performance of BOBYQA is also similar good, yet more function evaluations are required for both tests. Therefore, optimization of the shrinkage and warpage simulation in combination with the heat conduction simulation, using the BFGS is the best option of the tested algorithms. LINCOA did not show comparable good results, however, the algorithm's advantages of tackling constrained problems was not utilized in these tests. COBYLA shows an improved convergence for fewer design parameters, which underlines that this algorithm is better suited for smaller numbers of variables.

In the algorithm test, BFGS converged the fastest, while BOBYQA also showed quick convergence behavior. At the initial guess test (see Section 4.5), it was also shown that BFGS shows quicker convergence compared to BOBYQA. With the results from 4.6, it is shown that BOBYQA seems to be more robust than BFGS, since the latter fails to optimize the piecewise-linear *Hausdorff Distance*. However, since BFGS showed the least number of function evaluations, and since one function evaluation is time-expensive for larger simulations, we chose BFGS instead of BOBYQA. Nevertheless, further investigation is needed here, as the reduction of function evaluations is crucial for this optimization framework.

Siegbert et al. [31] compared different optimization algorithms for shape optimization of extrusion dies. The objective function used is based on the velocity distribution at the outflow. However, the results from this work show comparable findings. The authors obtained a fast convergence for the BFGS algorithm and showed even better results for BOBYQA. BOBYQA is also used for the shape optimization of extrusion dies in [14, 23].

5.2. Design Parameters and Sensitivity Analysis

The number and selection of design parameters have a significant impact on the shape optimization. The tests carried out show that a larger set of design parameters affect the results in two distinct ways. Firstly, the number of function evaluations increases as more parameters need to be considered. Secondly, the final value of the objective function may decrease, although such improvement cannot be guaranteed.
In cases where the system exhibits high symmetries, such as the cube, the objective function value remains the same, although the number of function evaluations is reduced.

The design parameter test (Section 4.3) and the sensitivity analysis (see Section 4.4) showed the strong impact of the number of design parameters on the optimization. Figure 20 illustrates that an increasing number of design parameters results in a lower objective function value. This leads to a massive increase of the number of function evaluations, although the improvement of the result converges. Therefore, the introduced sensitivity analysis in combination with a reduction of the design variables shows good results. With the threshold of the sensitivity according to the sensitivity's maximum in a certain spatial direction (see Section 3.4.3), the number of function evaluations could be decreased in all cases. For a fully symmetrical problem, the value of the objective function was not influenced negatively.

While a higher number of design parameters can lead to an improvement of the optimization result, which is reflected in a low value of the objective function, the associated computational time has to be considered, as it increases linearly with the number of design parameters. Since the L-Shape test showed a better result for a control mesh with a higher resolution but a lower number of design parameters, it can be derived that not only the absolute number of design parameters is important but also an adequate resolution of the control mesh. This yields in a crucial tradeoff between the computational cost, as we estimate, that for more complex shapes, such as undercuts or shapes with multiple corners, a higher set of design parameters is necessary for a sufficient shape optimization.

Testing the L-Shape with different numbers of design parameters showed that an improvement of the result can be achieved, by having a higher resolution control point mesh while having fewer design parameters in total. Hence, the number of design parameters is not the only factor in achieving satisfactory results. Instead, the resolution of the control point mesh in all spatial directions plays a vital role for shape optimization.

Determining the sensitivity of the design parameters depicted that it is possible to detect design parameters with a low impact on the shape deformation. By eliminating unsignificant design parameters, and therefore reducing the total number of design parameters, the number of function evaluations will decrease. However, this yields a less accurate solution of the optimization procedure, since fewer design parameters are processed. For this particular case of optimizing the L-Shape, a good trade-off between accuracy and number of function evaluations seems to be a value of $r = 0.025$.

5.3. Initial Guess

Using the initial guess as described in in Section 3.4.4 is clearly beneficial for the shape optimization for injection molding design. Final objective function values are significantly smaller (ca. $10\,\%$) and convergence behavior is affected positively as well. At the beginning of the optimization, the tests showed a peak in the objective function, which can be explained by an inaccurate approximation by the optimization algorithm. However, the proposed initial guess showed good results for all test cases (see Section 4.5) and worked well with BFGS as well as BOBYQA. It improves the optimization framework in two ways. First, the final value of the objective function is lower than compared to a test without the initial guess. Second, the number of function evaluations needed to reach convergence is decreased in all tests performed. Therefore, the use of this method is highly recommended for further use of the framework.

5.4. Objective Function

A suitable objective function is the key element for the IMO framework. Shape matching optimization method using the *Euclidean metric* was already performed in [11]. However, this work focused on shape matching techniques, using a weighted formulation of the *Euclidean distance* and the *Hausdorff Distance*. It is shown that the proposed *Distance Function* works fine in combination with the used algorithms. The *Hausdorff Distance* failed in combination within this framework. Even though it indicates the performance of the shape matching (compare Figure 28), some algorithms cannot handle the definition of the *Hausdorff Distance*.

The optimization with the *Hausdorff Distance* does not show satisfactory results. Only for BOBYQA and LINCOA the optimization converges to good solutions. Yet, the results are not as good as compared to results obtained by the *Distance Function*. A reason for the bad convergence behavior could be the piecewise linear definition of the *Hausdorff Distance* which is not continuously differentiable and is therefore difficult for the algorithms to be approximated by quadratic functions and that the *Hausdorff Distance* is quite sensitive to noise, as shown by [36].

5.5. Limitations

The proposed framework shows two major limitations: the dependency on the objective function and the dependency on the number of design parameters.

As discussed above, the objective function is the key element of the optimization. If it is not chosen properly, the algorithm will fail at optimizing or get stuck at a local minimum during optimization.

Computational cost is a limitation of this framework, as the number of design parameters may not be arbitrarily increased without incurring prohibitively high computational requirements. Therefore, the framework is limited to the number of design parameters.

Moreover, the result depends on the choice of the optimization algorithm as they can handle the minimization problem differently. For example, LINCOA does not show suitable results, whereas it might perform better for a constrained problem.

6. Application Case

In this section, a real-world example is optimized to demonstrate the applicability of the proposed method. The results from the previous sections are considered, such that we can see the overall procedure works for a more complex workpiece.

6.1. Test Case Description

Cable clamps, as seen in Figure 29, are a typical example for injection molding manufactured workpieces. Like many injection molded parts, they are often made of Polyamide Nylon, a thermoplastic with high stability and ductile strength. The material parameters used here, are given in Table 15.

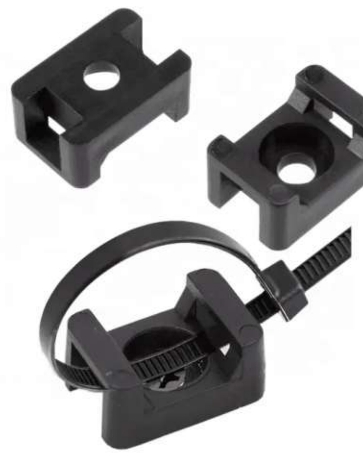


Figure 29: Polyamide Nylon Cable Clamps [1].

In order to reduce the computational time, the symmetrical properties of the clamp are used and only a quarter of the cable clamp is optimized. The clamp is divided at sections A and B, where we constrain the displacements in the normal direction of the plane. The geometry, obtained from [5], has the dimensions of $11.5 \times 8 \times 9\ mm$ and is and automatically meshed using HYPERMESH [18]. In this particular case, a tetrahedral mesh is used since NUTILS does not provide mesh import features for unstructured hexahedral meshes yet. We define the generated mesh as our reference mesh $\hat{\mathbf{m}}$, which is the targeted shape of the optimization. The reference mesh, as illustrated in Figure 30, consists of 14.892 tetrahedral elements, and therefore has 3.340 nodes. The initial temperature distribution obtained by the heat simulation is between 503 K inside of the clamp and 353 K at the surface.

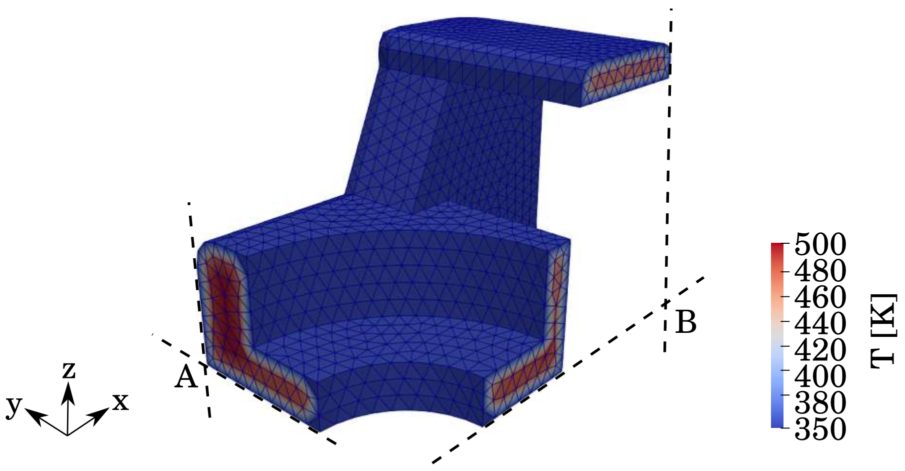


Figure 30: Reference mesh and initial temperature distribution. A and B inidicate the cutting surfaces.

6.2. Preliminary

The initial mesh $\tilde{\mathfrak{m}}$ for the optimization is evaluated by using the initial guess as defined in Section 3.4.4. Then, we define the control point mesh with a set of $4 \times 4 \times 4$ control points as seen in Figure 32, which yields a total set of 192 design parameters. Based on the initial guess and the control point mesh, a sensitivity analysis as described in Section 3.4.3 is performed.

In Figure 31, the determined sensitivities are illustrated as vectors, normalized with a *min-max-scale*, which means that the maximum values are normalized to one and the minimum values are normalized to zero. Here, the length of the arrow as well as the color describe the value of sensitivity. It can be observed that sensitivities are low at these points, where a control point is not in contact with the geometry, whereas highest sensitivities are determined at the *bridge* ($x = 11.5, z = 9$).

With the threshold of $r = 0.05$ from Equation 38, all design parameters with a magnitude of less than 5% than the maximum in the spatial direction i are eliminated. By applying this method, the number of design parameters is reduced to 175.

Figure 32 shows the control point mesh for the cable clamp. Control points marked in gray, have a reduced degree of freedom. They indicate, where the sensitivity analysis in combination with the reduction factor eliminated design parameters.

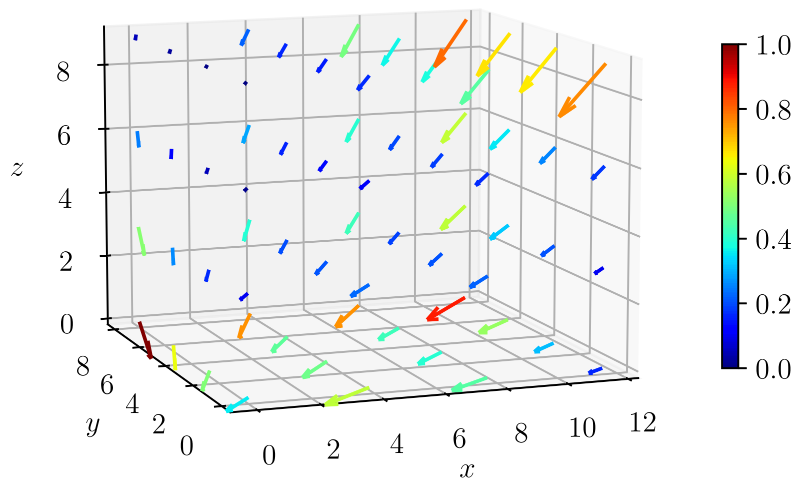


Figure 31: Field plot for the design parameter sensitivity of the cable clamp. The arrows show the direction of the sensitivity, length and color indicate the value (norm) of the sensitivity. All vectors are normalized with a *min-max-scale*.

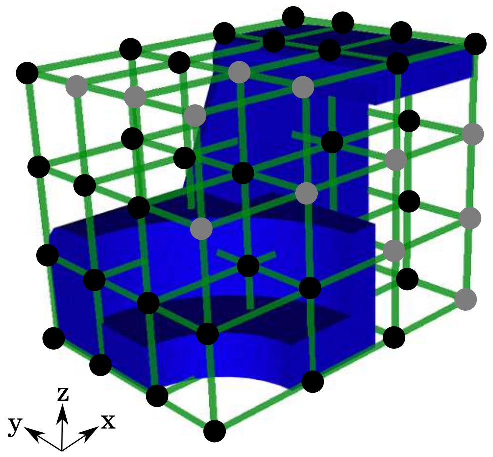


Figure 32: Control point mesh for the cable clamp. Gray points mark control points, with reduced DoF.

6.3. Results

For this optimization we employ BFGS with a tolerance limit of 10^{-4} and a targeted objective function value of 10^{-5}. The set of design parameter is generated based on the results from the sensitivity analysis using a threshold of $r = 0.05$.

After 9072 function evaluations, the optimization process stops with a final value of $1.00241 \cdot 10^{-5}$. Figure 33 shows the molding shape for the optimized cavity shape. The colors indicate the difference d between the coordinates of the molding shape and the reference shape. The largest difference between the compared geometries is observed at the upper right corner. Overall, the majority auf the surface shows small differences between the two meshes.

Figure 34 illustrates the trend of the objective Function during the optimization process. Initially, a peak occurs due to an inaccurate first approximation of the gradient. However, the objective function approaches the targeted value of 10^{-5}.

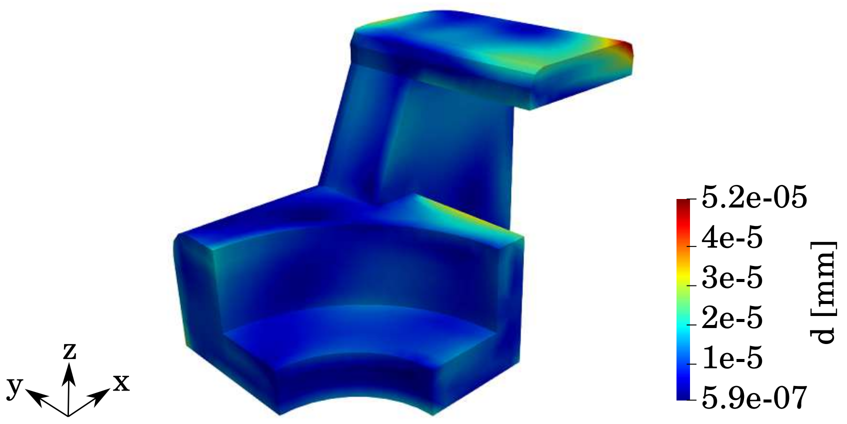


Figure 33: The shape of the mold after the simulation. The difference d (in mm) between the coordinates of the mold and the reference shape are colored.

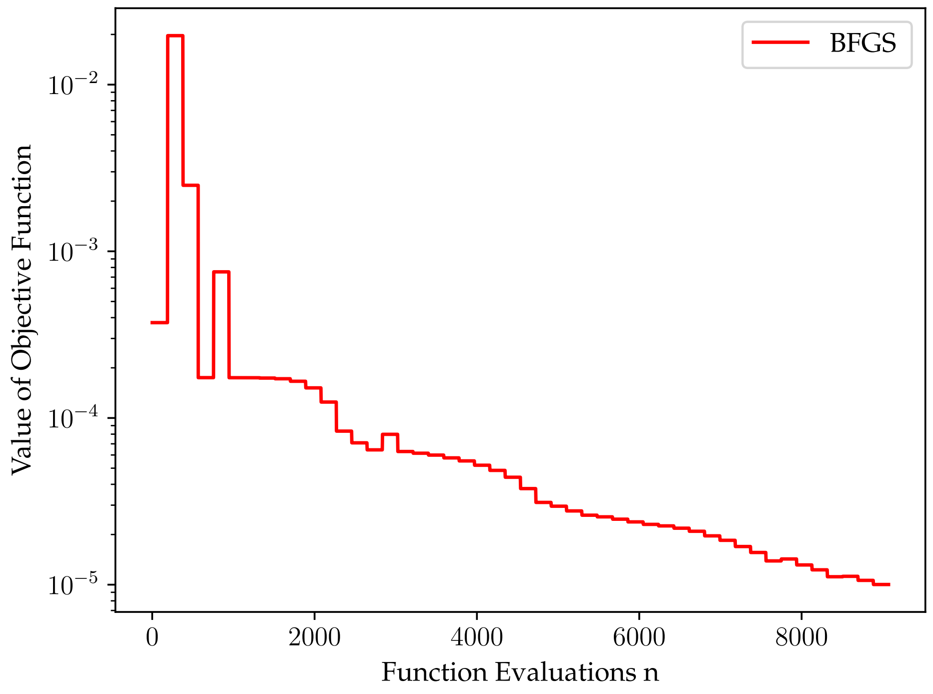


Figure 34: Plot of the objective function for the cable clamp optimization.

7. Conclusion and Outlook

7.1. Summary

The aim of this work was to determine the optimal design of a cavity shape for the injection molding process. Therefore, an object-oriented and PYTHON-based framework has been implemented, which combines the simulation of the injection molding process with a shape optimization.

The proposed optimization framework can be separated into four components. At first, the molding process is simulated by a heat conduction and shrinkage and warpage simulation with FEM, using an initial mesh as the shape of the cavity. Second, the resulting molding shape is transferred to an objective function, which measures the difference of the molding shape according to a reference shape. Then, the output of the objective function is passed to the optimization algorithm, which aims at minimizing the objective function. Finally, the algorithms input variables are used for a FFD of the cavity shape. The new shape yields as an input for the next function evaluation.

In order to show the functionality as well as the limitations of the model, the framework has been tested in multiple test cases.
Four different algorithms were examined, and it was shown that shape optimization for injection molding design works better for an optimization algorithm using a quadratic model approach. The best results are obtained with the use of BFGS, which showed the best performance in terms of needed function evaluations.
It could be observed that an increasing the number of design parameters results in a smaller value of the objective function at the end of the optimization. However, the tradeoff is a massive increase in function evaluations. This can be prevented by using a sensitivity analysis, which measures the influence of one single design parameter with respect to the objective function. Then, the set of design parameters can be reduced automatically by a threshold, which is beneficial in order to reduce the number of function evaluations.
Further, an initial guess for this framework is proposed, which adds the inverse displacements of a first simulation to the reference mesh. All tests showed an improvement due to this first guess.
Furthermore, two different objective functions were analyzed. The proposed *Distance Function* showed sufficient results for all cases. Optimizations using the *Hausdorff Distance* only worked sufficiently in combination with BOBYBA and is therefore not suitable for this framework.

Finally, a cable clamp, which yields as a real-world example, was optimized with the injection molding optimization framework. The results showed that the framework can handle more complex geometries in a suitable manner.

7.2. Outlook

By now, only one objective function works effectively within this framework. A possible next step could involve enhancing the proposed *Distance Function*, such as implementing a weighted objective function, which takes e.g. corners or important areas of the molding into account. Furthermore, exploring alternative measurements for shape matching techniques, as described in [36], could be evaluated in order to improve the process.

The proposed objective function is focused only on the minimization of shape differences. It could be easily adapted, in order to minimize residual stresses within the workpiece, while finding the optimal shape of the cavity. Also, it is possible to optimize certain material parameters combined with the structural shape optimization.

In order to use the framework more efficiently, further studies should be made to keep the number of needed function evaluations as low as possible. Therefore, the number of design parameters should remain small enough. One way to address this problem would be the introduction of a control mesh with an inhomogeneous distribution of design parameters. Also, a better understanding of the sensitivities of the design parameters would be advantageous in order to choose the threshold more effectively. Additionally, constraining the optimization problem, e.g. by constraining the design parameters to the symmetry of the problem could useful.

Furthermore, the simulation of the injection molding process requires further improvement. Currently, the process before the ejection of the part is modelled only by a transient heat conduction equation, while the process afterwards only considers nonlinear elastic material behavior.

A. Appendix

Table 12: Square test case simulation parameters.

ν [-]	α [1/K]	κ [W/(mK)]	Δt [s]	T_{in} [K]	T_{final} [K]
0.3	0.005	0.01	0.1	100	1

Table 13: Cube test case simulation parameters.

ν [-]	α [1/K]	κ [W/(mK)]	Δt [s]	T_{in} [K]	T_{final} [K]
0.3	0.05	0.2	0.1	100	1

Table 14: L-Shape test case simulation parameters.

ν [-]	α [1/K]	κ [W/(mK)]	Δt [s]	T_{in} [K]	T_{final} [K]
0.3	0.025	0.2	0.1	100	1

Table 15: Cable clamp simulation parameters.

ν [-]	α [1/K]	κ [W/(mK)]	Δt [s]	T_{in} [K]	T_{final} [K]
0.4 [22]	$15 \cdot 10^{-5}$ [22]	0.2 [22]	0.1	503 [8]	353 [8]

Table 16: Cube test case with 54 design parameters.

Index	DoF x	DoF y	DoF z
0	(-0.5, 0)	(-0.5, 0)	(-0.5, 0)
1		(-0.5, 0)	(-0.5, 0)
2	(0, 0.5)	(-0.5, 0)	(-0.5, 0)
3	(-0.5, 0)		(-0.5, 0)
4			(-0.5, 0)
5	(0, 0.5)		(-0.5, 0)
6	(-0.5, 0)	(0, 0.5)	(-0.5, 0)
7		(0, 0.5)	(-0.5, 0)
8	(0, 0.5)	(0, 0.5)	(-0.5, 0)
9	(-0.5, 0)	(-0.5, 0)	
10		(-0.5, 0)	
11	(0, 0.5)	(-0.5, 0)	
12	(-0.5, 0)		
14	(0, 0.5)		
15	(-0.5, 0)	(0, 0.5)	
16		(0, 0.5)	
17	(0, 0.5)	(0, 0.5)	
18	(-0.5, 0)	(-0.5, 0)	(0, 0.5)
19		(-0.5, 0)	(0, 0.5)
20	(0, 0.5)	(-0.5, 0)	(0, 0.5)
21	(-0.5, 0)		(0, 0.5)
22			(0, 0.5)
23	(0, 0.5)		(0, 0.5)
24	(-0.5, 0)	(0, 0.5)	(0, 0.5)
25		(0, 0.5)	(0, 0.5)
26	(0, 0.5)	(0, 0.5)	(0, 0.5)

References

[1] Cable clamps. https://s.alicdn.com/@sc04/kf/Hd2ab771eb2034bfca1c74a57c4aa3443b.png_960x960.png, 2023. Accessed: 2023-06-14.

[2] Helmut Alt. The computational geometry of comparing shapes. *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 235–248, 2009.

[3] Steven E Benzley, Ernest Perry, Karl Merkley, Brett Clark, and Greg Sjaardama. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings, 4th international meshing roundtable*, volume 17, pages 179–191. Sandia National Laboratories Albuquerque, NM, 1995.

[4] Bruce Catoen and Herbert Rees. *Injection Mold Design Handbook*. Carl Hanser Verlag GmbH Co KG, 2021.

[5] Dmitrii Charuiskii. Cable clamp stm-2. https://grabcad.com/library/cable-clamp-stm-2-1, 2023. Accessed: 2023-06-14.

[6] Philippe G Ciarlet. *Three-dimensional elasticity*. Elsevier, 1988.

[7] AO Cifuentes and A Kalbag. A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis. *Finite Elements in Analysis and Design*, 12(3-4):313–318, 1992.

[8] Ecolmolding Co. Nylon injection molding. https://www.ecomolding.com/nylon-injection-molding/, 2023. Accessed: 2023-06-14.

[9] J Austin Cottrell, Thomas JR Hughes, and Yuri Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[10] Carl De Boor. On calculating with b-splines. *Journal of Approximation theory*, 6(1):50–62, 1972.

[11] Maya De Buhan, Charles Dapogny, Pascal Frey, and Chiara Nardoni. An optimization method for elastic shape matching. *Comptes Rendus Mathematique*, 354(8):783–787, 2016.

[12] Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. *Finite element method*. John Wiley & Sons, 2012.

[13] Stefanie Elgeti. Skript zur veranstaltung einführung in die finite elemente methode. Lecture Notes, 2020.

[14] Stefanie Elgeti, Markus Probst, Christian Windeck, Marek Behr, W Michaeli, and Ch Hopmann. Numerical shape optimization as an approach to extrusion die design. *Finite Elements in Analysis and Design*, 61:35–43, 2012.

[15] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[16] Raphael T. Haftka and Jaroslaw Sobieszczanski-Sobieski. pages 3834–3836. Springer US, Boston, MA, 2009. ISBN 978-0-387-74759-0. doi: 10.1007/978-0-387-74759-0_669. URL https://doi.org/10.1007/978-0-387-74759-0_669.

[17] Gerhard A Holzapfel. Nonlinear solid mechanics: a continuum approach for engineering science, 2002.

[18] Altair Engineering Inc. Hypermesh. https://2022.help.altair.com/2022.1/hwdesktop/hm/index.htm. Accessed: 2023-06-25.

[19] Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.

[20] Alexander Lion. On the large deformation behaviour of reinforced rubber at different temperatures. *Journal of the Mechanics and Physics of Solids*, 45(11-12):1805–1834, 1997.

[21] Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Shape optimization for viscous flows by reduced basis methods and free-form deformation. *International Journal for Numerical Methods in Fluids*, 70(5):646–670, 2012.

[22] Matweb. Overview of materials for nylon 66.6. https://www.matweb.com/search/datasheet_print.aspx?matguid=26386631ec1b49eeba62c80a49730dc4, 2023. Accessed: 2023-06-14.

[23] L Pauli, M Behr, and S Elgeti. Towards shape optimization of profile extrusion dies with respect to homogeneous die swell. *Journal of Non-Newtonian Fluid Mechanics*, 200:79–87, 2013.

[24] Michael JD Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[25] Michael JD Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 26, 2009.

[26] Michael JD Powell. On fast trust region methods for quadratic models with linear constraints. *Mathematical Programming Computation*, 7:237–267, 2015.

[27] Tom M Ragonneau and Zaikun Zhang. Pdfo–a cross-platform package for powell's derivative-free optimization solver. *arXiv preprint arXiv:2302.13246*, 2023.

[28] Jamshid Samareh. Aerodynamic shape optimization based on free-form deformation. In *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 4630, 2004.

[29] Axel Schumacher. *Optimierung mechanischer strukturen*. Springer, 2020.

[30] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, 1986.

[31] Roland Siegbert, Johannes Kitschke, Hatim Djelassi, Marek Behr, and Stefanie Elgeti. Comparing optimization algorithms for shape optimization of extrusion dies. *PAMM*, 14(1):789–794, 2014.

[32] Mario Studer and Frank Ehrig. Numerical shape optimization as an approach to reduce material waste in injection molding. *The International Journal of Advanced Manufacturing Technology*, 78:1557–1571, 2015.

[33] Laurens van Lieshout. Injection moulding process. https://commons.wikimedia.org/wiki/File:Injection_moulding_process.png, 2007. Accessed: 2023-04-25.

[34] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[35] J.S.B. van Zwieten, G.J. van Zwieten, and W. Hoitinga. Nutils 7.0, 2022.

[36] Remco C Veltkamp. Shape matching: Similarity measures and algorithms. In *Proceedings International Conference on Shape Modeling and Applications*, pages 188–197. IEEE, 2001.

[37] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[38] Francis Williams. Point cloud utils, 2022. https://www.github.com/fwilliams/point-cloud-utils.

[39] Stephen J. Wright. Encyclopedia britannica. https://www.britannica.com/science/optimization. Accessed: 2023-04-17.

[40] Yingjie Xu, QingWen Zhang, Weihong Zhang, and Pan Zhang. Optimization of injection molding process parameters to improve the mechanical performance of polymer product against impact. *The International Journal of Advanced Manufacturing Technology*, 76:2199–2208, 2015.

[41] Florian Zwicke. *Inverse shape design in injection molding based on the finite element method*. PhD thesis, Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2020, 2020.

[42] Florian Zwicke and Stefanie Elgeti. Inverse design based on nonlinear thermoelastic material models applied to injection molding. *Finite Elements in Analysis and Design*, 165:65–76, 2019.