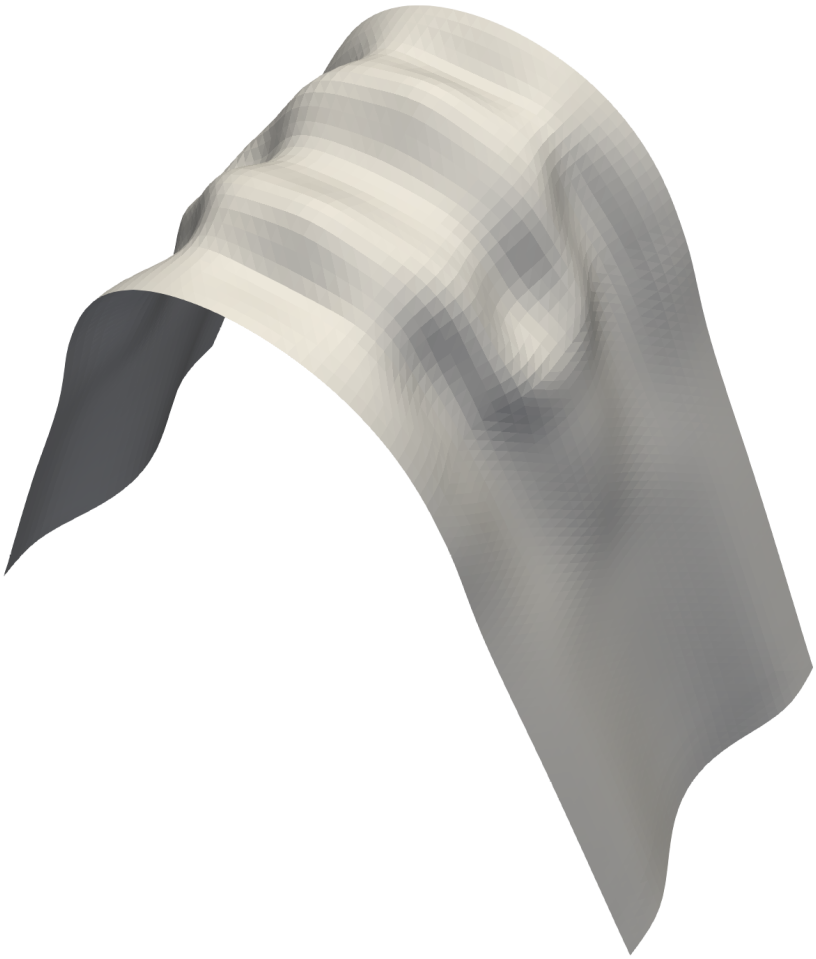
Statik



MASTER THESIS

Robust Node-Based Shape Optimization with Vertex Morphing

Philipp Jakobs

Lehrstuhl für Statik
Prof. Dr.-Ing. K.-U. Bletzinger
Technische Universität München

Robust Node-Based Shape Optimization with Vertex Morphing

Submitted to the Department of Civil, Geo and Environmental Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. Compuational Mechanics
at the Technical University of Munich.

Supervised by	David Schmölz, Bastian Devresse, Anoop Kodakkal
	Prof. Dr.-Ing. Kai-Uwe Bletzinger
	Chair of Structural Analysis
Submitted by	Philipp Jakobs
	Steinhauser Str. 37
	81677 München
	Student number: 31790512
Submitted on	Munich, 14.10.2024

Abstract

This thesis focuses on the development of a robust node-based shape optimization framework that integrates vertex morphing and uncertainty quantification (UQ) techniques to improve the design of engineering structures. Traditional shape optimization methods are often limited by their inability to account for uncertainties in real-world applications, such as variations in material properties, boundary conditions, and external forces. This research addresses this limitation by combining node-based optimization with two uncertainty quantification methods: Monte Carlo simulations and Polynomial Chaos Expansion (PCE). This ensures the robustness of the design under stochastic variations. To overcome the challenges of node-based optimization, vertex morphing is used.

The proposed methodology is validated through academic case studies, including an arch structure subjected to random load conditions. These examples demonstrate the framework's capability to produce designs that are both optimal under nominal conditions and resilient to uncertainties. The results confirm the effectiveness of the approach in delivering reliable, high-performance designs that account for real-world variability.

Keywords

FEM; Robust Optimization; Uncertainty Quantification; Polynomial Chaos Expansion, Monte Carlo, Vertex Morphing

Kurzfassung

Diese Masterarbeit beschäftigt sich mit der Entwicklung eines robusten, knotenbasierten Formoptimierungsframeworks, das Vertex-Morphing und Methoden der Unsicherheitsquantifizierung (UQ) vereint, um das Design von technischen Strukturen zu verbessern. Traditionelle Methoden der Formoptimierung stoßen oft an ihre Grenzen, da sie Unsicherheiten in realen Anwendungen, wie Materialschwankungen, variierende Randbedingungen oder externe Kräfte, nicht ausreichend berücksichtigen. Diese Forschung adressiert diese Problematik, indem sie knotenbasierte Optimierung mit zwei Methoden der Unsicherheitsquantifizierung kombiniert: Monte-Carlo-Simulationen und Polynomial Chaos Expansion (PCE). Dadurch wird sichergestellt, dass das Design auch unter stochastischen Einflüssen robust bleibt. Um die Herausforderungen der knotenbasierten Optimierung zu überwinden, wird Vertex Morphing als Regularisierungsmethode eingesetzt.

Die vorgeschlagene Methodik wird anhand akademischer Fallstudien validiert, darunter eine Bogenstruktur, die zufälligen Lastbedingungen ausgesetzt ist. Diese Beispiele zeigen die Fähigkeit der Methode, Designs zu erzeugen, die sowohl unter nominalen Bedingungen optimal sind als auch widerstandsfähig gegenüber Unsicherheiten. Die Ergebnisse bestätigen die Wirksamkeit des Ansatzes bei der Entwicklung zuverlässiger, leistungsstarker Designs, die reale Variabilitäten berücksichtigen.

Schlüsselwörter

FEM; Robuste Optimierung; Unsicherheitsquantifizierung, Polynomial Chaos Expansion, Monte Carlo, Vertex Morphing

Table of Contents

1 Introduction

1.1 Robust Node-based Shape Optimization

Shape optimization is a fundamental tool in the design of engineering structures across various fields, aiming to achieve optimal geometries that meet specific performance criteria, such as minimizing drag in fluid flow or maximizing structural stiffness. Traditionally, shape optimization has relied on Computer-Aided Design (CAD) parameterization, which offers robustness by preserving geometrical quality. However, CAD-based methods often suffer from a limited design space, constrained by predefined parameters.

To overcome this limitation, node-based shape optimization has emerged as an alternative. In this approach, mesh node positions are directly manipulated, providing a larger design space and enabling more flexible and often non-intuitive shape modifications. Despite this flexibility, node-based methods pose significant challenges, including mesh dependency, potential shape irregularities, and the handling of a large number of design variables [1]. To address these issues, regularization techniques such as vertex morphing, introduced by Hojjat et al. [2], have been developed. Vertex morphing integrates mesh filtering and smoothing directly into the optimization process, allowing the management of complex design spaces while ensuring smooth geometric transitions.

Another key challenge in shape optimization is the need to account for uncertainties inherent in real-world applications, such as variations in material properties, boundary conditions, and external forces. Designs optimized under nominal conditions may perform poorly when subjected to slight perturbations. Robust optimization methodologies aim to mitigate this risk by ensuring that designs perform consistently across a range of uncertain conditions. While robust optimization has been increasingly applied to CAD-based shape optimization, particularly in aerodynamic applications (see e.g., [3], [4], [5]), its use in node-based shape optimization remains largely unexplored.

This thesis focuses on developing a robust node-based shape optimization framework that combines Polynomial Chaos Expansion (PCE) as an Uncertainty Quantification (UQ) tool with vertex morphing as a parameterization technique.

1.2 Outline

This thesis is organized to explore robust shape optimization under uncertainty, progressing from foundational concepts to the development of a comprehensive optimization framework. Chapter 2 introduces the core principles and mathematical formulations of shape optimization, including methods of shape representation, objective functions, constraints, and gradient-based algorithms. Chapter 3 introduces the fundamentals of UQ, with a focus on Monte Carlo simulations and PCE as key techniques for addressing uncertainties. In Chapter 4, these concepts are integrated to develop a robust optimization framework. Chapter 5 applies this framework to practical case studies, demonstrating its effectiveness in two academic examples. Finally, Chapter 6 summarizes the research findings.

2 Shape Optimization

In this chapter, the fundamentals of shape optimization will be discussed. As many different methods and techniques exist, the differences are explained and brought together, illustrating the final chosen workflow: A node-based shape optimization algorithm that uses adjoint sensitivities and the vertex morphing technique for regularization.

2.1 Optimization Problem

2.1.1 Standard optimization problem

Shape optimization begins with the formulation of a mathematical optimization problem, where the goal is to find the optimal shape of a structure or component that minimizes (or maximizes) a specific objective function. The optimization problem can be formulated in a standard format

$$\begin{aligned} \textbf{Minimize:} \quad & f(\mathbf{x}) \\ \textbf{Design variables:} \quad & \mathbf{x} \\ \textbf{Subject to:} \quad & g_i(\mathbf{x}) \leq 0, \\ & h_j(\mathbf{x}) = 0, \\ & r_k(\mathbf{x}, \mathbf{u}(\mathbf{x})) = 0 \end{aligned} \tag{2.1}$$

where f is an objective function that has to be minimized by varying the design variables $\mathbf{x}$, such that the equality constraints h_j and the inequality constraints g_k are satisfied. $\mathbf{u}$ is the vector of state variables and r_k are the residuals of the state governing equations, i.e., the physical constraints on the problem.

2.1.2 Design variables

The design variables are represented by the column vector $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ and define a given design. The size of the vector n determines the dimensionality. In structural optimization, we typically classify our optimization problem depending on the choice of design variables.

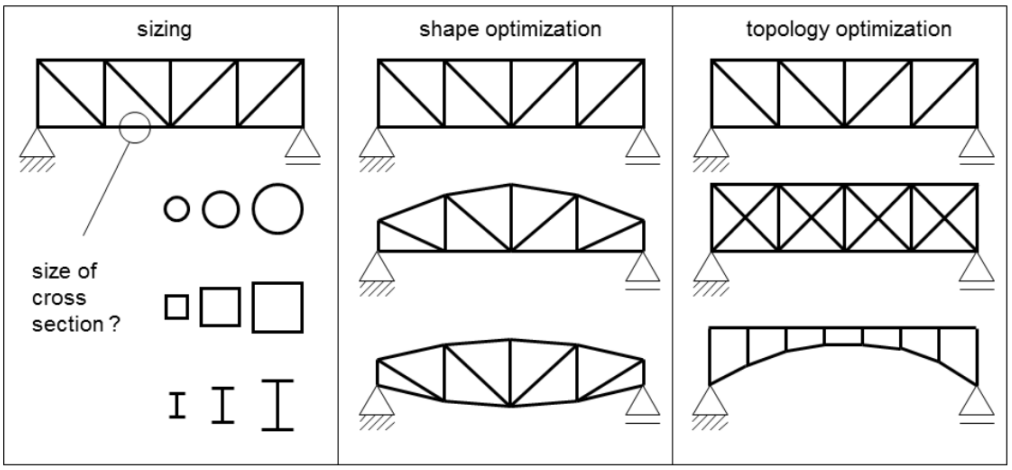


Figure 1 Levels of structural optimization [6]

Example The differences between the different classes can be made clear with the help of a truss structure. In *sizing* both the shape and the connectivity of the truss members are fixed. The strategy would be to adjust the thickness or cross-sectional areas of each member to optimize for a lighter structure. In *topology optimization*, via the creation or removal of members, the connectivity and layout of the structure would be explored to find an optimal structure. In *shape optimization*, the geometry of the structure, i.e., the position of each joint, has to be determined, while the connectivity is given. Fig. 1 demonstrates these differences.

Whilst for shape optimization of 1D truss structures, the vector of design variables consists of the truss joint coordinates, different types of approaches have been developed for continuous 2D and 3D models. The main two methods are node-based (parameter-free) and CAD-based (parametric) approaches. Node-based shape optimization involves directly manipulating the nodes within a finite element mesh to achieve desired performance improvements, offering fine-tuned control over local geometrical features but often at the cost of increased computational complexity. Conversely, CAD-based shape optimization adjusts the geometric parameters of the CAD model itself, preserving design intent and ensuring that changes remain consistent with manufacturable and aesthetic considerations. This method is particularly advantageous during early design stages, where broader changes to fundamental geometric features are required. Understanding the strengths and limitations of each approach is crucial for selecting the appropriate optimization strategy based on the specific requirements and stage of the design process.

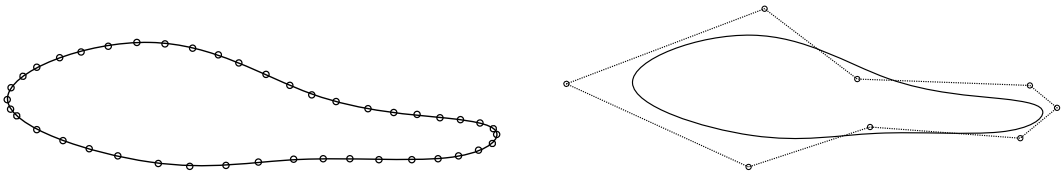


Figure 2 Node-based (left) vs CAD-based (right) optimization of a wing cross-section, adapted from [7]

In this thesis, a **node-based** shape optimization workflow is chosen.

2.1.3 Objective and constraint functions

In shape optimization, both objective functions f and constraints g, h are expressed as response functions J that describe how the shape of a design influences its performance. The most common examples for strucural responses used as objective functions or constraints include stresses, strain energy and weight. Further constraints could include the goal of maintaining a specific volume or shape restrictions. These constraints ensure that the optimized shape meets necessary design criteria, such as structural integrity or manufacturing limitations.

2.1.4 Solution strategies

Zero-order methods

Zero-order methods, also known as derivative-free optimization methods, are techniques used for optimizing functions without requiring gradient information. They are particularly useful when the objective function is non-differentiable, noisy, or computationally expensive to evaluate. Search methods, such as the Nelder-Mead simplex algorithm [8], systematically explore the search space by evaluating function values and iteratively refining the solution. Population-based methods, including genetic algorithms [9], evolutionary strategies [10] and particle swarm optimization [11], leverage a population of potential solutions that evolve over iterations to explore the search space more broadly and avoid local optima. These methods are essential tools for solving complex optimization problems where traditional gradient-based techniques are not applicable.

Gradient-based methods

Gradient-based methods on the other hand are optimization techniques that utilize gradient information to find the minimum or maximum of a function. These methods are divided into first-order and second-order approaches. First-order methods, such as Gradient Descent and its variants, use only the first derivative (gradient) to guide the search direction, making them computationally efficient and suitable for large-scale problems. In contrast, second-order methods, like Newton's method [12] and the BFGS algorithm [13], employ both the gradient and the Hessian matrix (second derivative) to determine the search direction and step size, providing faster convergence and greater accuracy for problems where the Hessian is computationally feasible to compute.
In this thesis, we will employ two first-order methods, which will be explained in the next section.

2.2 Optimization Algorithms

2.2.1 Steepest Descent

Steepest descent is a first-order optimization method to find the minimum of unconstrained problems. The method iteratively updates the solution by moving in the direction of the negative gradient, which represents the steepest descent direction. The update rule for the iteration k is given by:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha^{(k)} \nabla f(\mathbf{x}^{(k)}), \tag{2.2}$$

where $\alpha^{(k)}$ is the step size, and $\nabla f(\mathbf{x}^{(k)})$ is the gradient of f at $\mathbf{x}^{(k)}$. The steepest descent method is computationally simple and effective for many problems, though it may converge slowly if the step size is not properly chosen.

2.2.2 Gradient projection

Gradient projection is a type of algorithm that was introduced by Rosen [14] in 1960 to solve constrained problems. We start with the optimization problem:

$$\begin{aligned} \textbf{Minimize:} \quad & f(\mathbf{x}) \\ \textbf{Design variables:} \quad & \mathbf{x} \\ \textbf{Subject to:} \quad & g_i(\mathbf{x}) \leq 0, \\ & h_j(\mathbf{x}) = 0, \\ & r_k(\mathbf{x}, \mathbf{u}(\mathbf{x})) = 0 \end{aligned} \tag{2.1 revisited}$$

If we select only the r active constraints, we can define an $n \times r$ matrix $\mathbf{N}$, where the columns of this matrix are the gradients of the active constraints. The basic assumption of the gradient projection method is that $\mathbf{x}$ lies on the tangential subspace to the boundary of the active constraints. If our solutions $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$ at iteration k and $k+1$ satisfy the constraints, then the constraints can be rewritten as:

$$\mathbf{N}^T \mathbf{s} = 0 \tag{2.3}$$

where s is a search direction. To project the steepest descent direction $-\nabla f$ onto the tangent

subspace of the active set of constraints, we can redefine the problem as follows:

$$\text{minimize:} \quad \mathbf{s}^T \nabla f \tag{2.4}$$

$$\text{s.t.:} \quad \mathbf{N}^T \mathbf{s} = 0, \tag{2.5}$$

$$\mathbf{s}^T \mathbf{s} = 1 \tag{2.6}$$

The second condition bounds the solution. The Lagrangian function for this problem is:

$$L(s, \lambda, \mu) = \mathbf{s}^T \nabla f - \mathbf{s}^T \mathbf{N} \lambda - 2\mu(\mathbf{s}^T \mathbf{s} - 1) \tag{2.7}$$

where λ and μ are called Lagrange multipliers. The condition for L to be stationary is:

$$\frac{\partial L}{\partial \mathbf{s}} = \nabla f - \mathbf{N}\lambda - 2\mu \mathbf{s} = 0 \tag{2.8}$$

We can find λ by multiplying (2.8) by $\mathbf{N}^T$ and using the condition from (2.3):

$$\lambda = (\mathbf{N}^T \mathbf{N})^{-1} \mathbf{N}^T \nabla f \tag{2.9}$$

The feasible search direction $\mathbf{s}$ is given by:

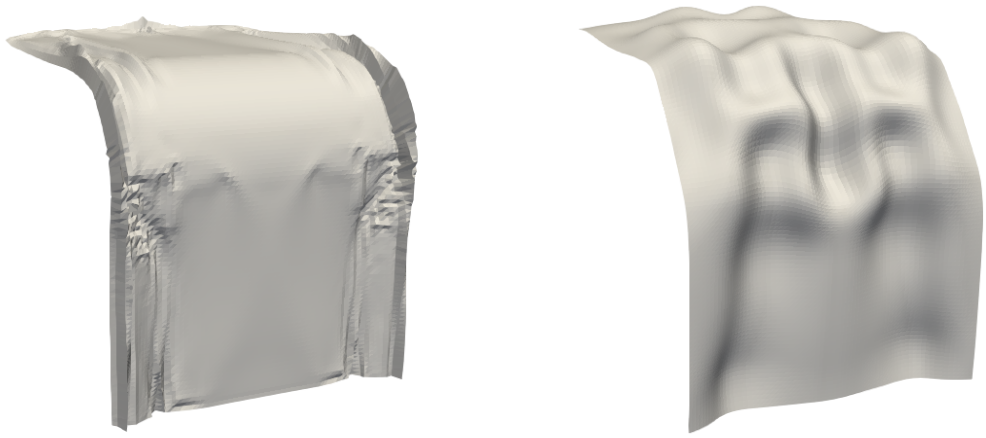
$$\mathbf{s} = \frac{1}{2\mu} [I - \mathbf{N}(\mathbf{N}^T \mathbf{N})^{-1} \mathbf{N}^T] \nabla f \tag{2.10}$$

After finding the feasible search direction, a new shape $\mathbf{x}^{(k+1)}$ can be found. A line-search can be applied to find the step size $\alpha^{(k)}$ that sufficiently reduces the objective function, or a constant step size can be used. The design update is then given by:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{s} \tag{2.11}$$

2.3 Regularization using Vertex Morphing

Before we can apply any of the solution strategies to our design variables, we need to introduce a regularization step. Regularization in node-based shape optimization is essential to manage the large number of design solutions that can arise when each node in the control mesh acts as a design parameter. Without regularization, many of these solutions tend to be noisy, resulting in non-smooth design surfaces and low-quality meshes. These irregularities can lead to unusable or invalid shapes, such as meshes with self-intersections, kinks, or other artifacts that do not satisfy the desired constraints. Regularization techniques, like filtering operations on shape gradients, help produce smooth and valid shape updates that can be applied to the mesh without causing numerical issues. By ensuring a minimum level of smoothness, regularization enhances the stability and practicality of the optimization process.



(a) Node-based optimization without regularization

(b) Node-based optimization with vertex morphing

Figure 3 Motivation for regularization: Directly using the nodal sensitivities leads to undesired artifacts (a), whereas applying a regularization technique leads to smoother geometries (b).

One regularization method is the vertex morphing technique as first presented by Hojjat et al. [2]. The main idea of this method is to introduce a surface control field and define an explicit mapping between the geometry space and control space. After a backward mapping, the optimization problem is solved in the control space, and the shape update is finally obtained after a forward mapping.

Consider a body occupying a volume Ω with a surface Γ. Material points on Γ are described using surface coordinates $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_{d-1}\} \in \mathbb{R}^{d-1}$. For a 3-dimensional geometry, $d = 3$. Each material point on the surface can be traced through the spatial position vector field $\mathbf{x}(\boldsymbol{\theta}, t) \in \mathbb{R}^3$, expressed as:

$$\mathbf{x}(\boldsymbol{\theta}, t) = \begin{pmatrix} x_1(\mathbf{p}(\boldsymbol{\theta}, t)) \\ x_2(\mathbf{p}(\boldsymbol{\theta}, t)) \\ x_3(\mathbf{p}(\boldsymbol{\theta}, t)) \end{pmatrix} \tag{2.12}$$

Here, the pseudo-time t represents the iterative evolution of the geometry during the optimization process.

2.3.1 Control Field and Kernel-Based Filter

The proposed control field $\mathbf{p}(\boldsymbol{\theta}, t) \in \mathbb{R}^3$, which depends on the surface coordinates, governs the geometry through a functional $A(\mathbf{p}, \boldsymbol{\theta}) = \mathbf{x}(\boldsymbol{\theta})$, where $A : \mathbb{R}^3 \to \mathbb{R}^3$. The relationship between $\mathbf{p}(\boldsymbol{\theta}, t)$ and $\mathbf{x}(\boldsymbol{\theta}, t)$ is defined by an explicit kernel-based filter:

$$\mathbf{x}(\boldsymbol{\theta}_0) = A(\mathbf{p}, \boldsymbol{\theta}_0) = \int_\Gamma A(\boldsymbol{\theta} - \boldsymbol{\theta}_0)\mathbf{p}(\boldsymbol{\theta})\, d\Gamma \tag{2.13}$$

The coordinates $\mathbf{x}$ at point $\boldsymbol{\theta}_0$ are obtained as a convolution of the operator $A(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$ over the control field $\mathbf{p}(\boldsymbol{\theta})$. The function A serves as a kernel, typically a linear hat function or a Gaussian function. Both types of filters produce similar qualitative results. The Gaussian function is defined as:

$$A(\boldsymbol{\theta} - \boldsymbol{\theta}_0) = \frac{1}{r\sqrt{2\pi}} e^{-\frac{\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2}{2r^2}} \tag{2.14}$$

The linear hat function is defined as:

$$A(\boldsymbol{\theta} - \boldsymbol{\theta}_0) = \begin{cases} 1 - \frac{\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|}{r}, & \text{if } \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\| \leq r, \\ 0, & \text{if } \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\| > r, \end{cases} \tag{2.15}$$

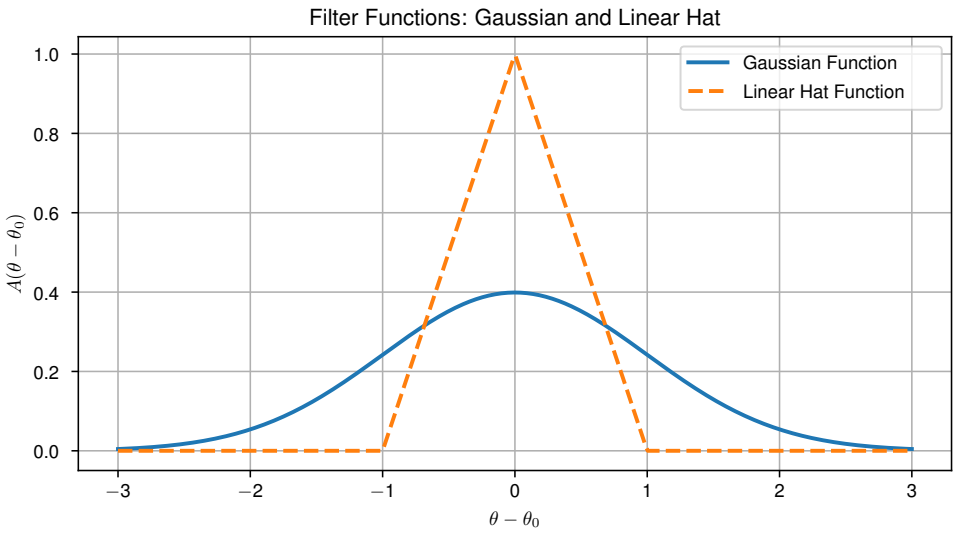


Figure 4 Two commonly used filter functions: Gaussian and linear hat

From Eq. 2.13, the derivative of the spatial vector $\mathbf{x}$ at point $\boldsymbol{\theta}_0$ with respect to the control coordinates at point $\boldsymbol{\theta}_1$ is:

$$\frac{d\mathbf{x}(\boldsymbol{\theta}_0)}{d\mathbf{p}(\boldsymbol{\theta}_1)} = A(\boldsymbol{\theta}_1, \boldsymbol{\theta}_0) \tag{2.16}$$

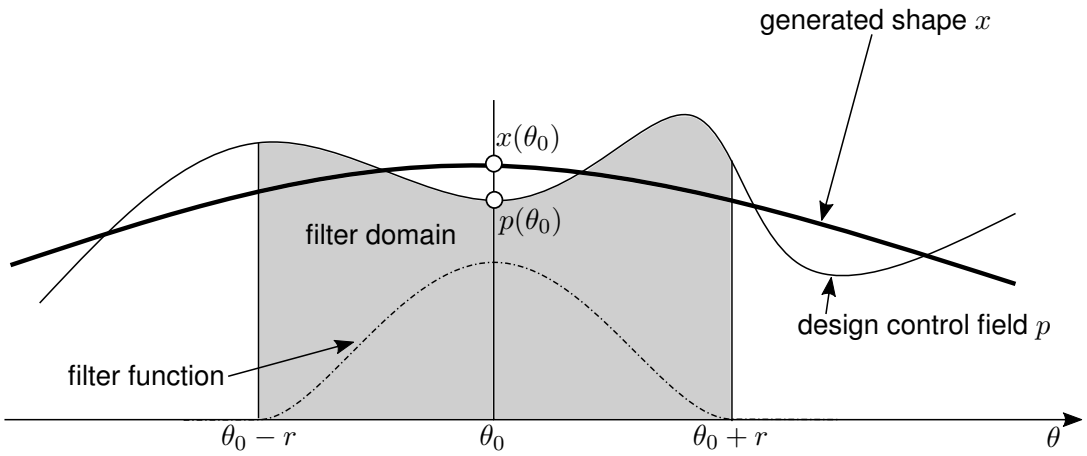


Figure 5 Filtering of design control field to generate shape, adapted from [15]

Fig. 5 shows the basic idea of the application of the control field in 1D.

2.3.2 Objective Function Formulation

We consider a design objective Φ with a volume density f_Ω for minimization. The optimization takes place in the control space, so the objective is a function of the control field:

$$\Phi(\mathbf{p}(\boldsymbol{\theta}, t)) = \int_\Omega f_\Omega(\mathbf{p}(\boldsymbol{\theta}, t))\, d\Omega \tag{2.17}$$

Or, expressed as a surface Γ problem, as is most often the case in shape optimization:

$$\Phi(\mathbf{p}(\boldsymbol{\theta}, t)) = \int_\Gamma f_\Gamma(\mathbf{p}(\boldsymbol{\theta}, t))\, d\Gamma \tag{2.18}$$

The surface density f_Γ is obtained from the volume density f_Ω by integration over the thickness for shell geometries:

$$f_\Gamma = \int_h f_\Omega\, dn; \quad \mathbf{n}(\boldsymbol{\theta}, t) \in \mathbb{R}^3 \tag{2.19}$$

2.3.3 Discretization of the Problem

To solve the continuous problem, it must be discretized. The geometry field $\mathbf{x}(\boldsymbol{\theta})$ is discretized into finite parts with shape functions R_j and nodal coordinates $\mathbf{X} = \{X_1, X_2, \ldots, X_n\} \in \mathbb{R}^n$:

$$\mathbf{x}(\boldsymbol{\theta}) \approx \mathbf{x}^h(\boldsymbol{\theta}) = \sum_i R_i(\boldsymbol{\theta}) X_i \tag{2.20}$$

Similarly, the design control field $\mathbf{p}(\boldsymbol{\theta})$ is discretized with control points p_j and shape functions N_j:

$$\mathbf{p}(\boldsymbol{\theta}, t) \approx \mathbf{p}^h(\boldsymbol{\theta}) = \sum_j N_j(\boldsymbol{\theta}) p_j(t) \tag{2.21}$$

Eq. 2.13 transforms to:

$$\mathbf{x}^h(\boldsymbol{\theta}_0) = \int_\Gamma A(\boldsymbol{\theta} - \boldsymbol{\theta}_0) N_j(\boldsymbol{\theta}) p_j \, d\Gamma \tag{2.22}$$

The control points p_j are independent of the integral and can be factored out:

$$\mathbf{x}^h(\boldsymbol{\theta}_0) = \left(\int_\Gamma A(\boldsymbol{\theta} - \boldsymbol{\theta}_0) N_j(\boldsymbol{\theta}) \, d\Gamma \right) p_j = B_j(\boldsymbol{\theta}_0) p_j \tag{2.23}$$

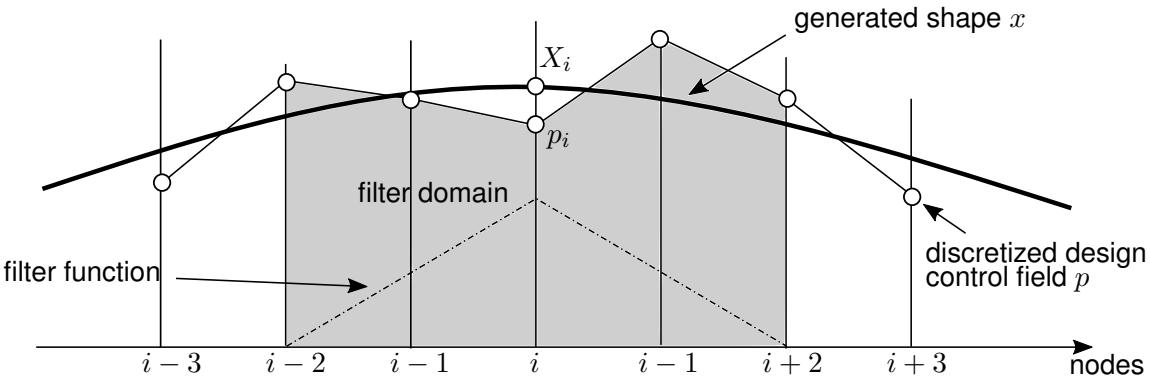


Figure 6 Linearly approximated design control field, adapted from [15]

2.3.4 Optimization Process

The optimization design variables are the control points p_j of the geometrical coordinates X_j. The update rule for the new design according to the steepest descent method described in Section 2.2.1 is:

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} + \Delta\mathbf{p}^{(k)} = \mathbf{p}^{(k)} - \alpha^{(k)} \frac{df_\Gamma}{d\mathbf{p}^{(k)}} \tag{2.24}$$

The variation $\delta\Phi$ due to an infinitesimal variation of the geometry $\delta\mathbf{x}$ at the surface Γ is:

$$\delta\Phi = \int_\Gamma \frac{df_\Gamma}{d\mathbf{x}} \delta\mathbf{x} \, d\Gamma \tag{2.25}$$

In the control space, this is approximated as δF:

$$\delta\Phi(\mathbf{p}) \approx \delta F(\mathbf{p}^h) = \frac{dF}{dp_i}\delta p_i = \int_\Gamma \frac{df_\Gamma}{d\mathbf{p}} N_i \delta p_i \, d\Gamma \tag{2.26}$$

The design gradient at position $\boldsymbol{\theta}_1$ is:

$$\frac{df_\Gamma}{d\mathbf{p}}(\boldsymbol{\theta}_1) = \int_\Gamma A^{\mathsf{adj}}(\boldsymbol{\theta}_1 - \boldsymbol{\theta}) \frac{df_\Gamma}{d\mathbf{x}} \, d\Gamma \tag{2.27}$$

The shape gradient field $\frac{df_\Gamma}{d\mathbf{x}}$ also needs to be discretized. Applying the geometry discretization yields:

$$\frac{df_\Gamma}{d\mathbf{x}^h} \approx \sum_i R_i(\boldsymbol{\theta}) \frac{df_\Gamma}{dX_i} \tag{2.28}$$

The term $\frac{df_\Gamma}{dX_i}$ is calculated locally for each nodal coordinate by a sensitivity analysis from a solver. The design gradient $\frac{df_\Gamma}{dp_j}$ is computed from these values.

Similarly, the nodal constraint gradients are calculated using the same approach. The gradient of a constraint function g with respect to the nodal coordinates is given by:

$$\frac{dg_\Gamma}{d\mathbf{x}^h} \approx \sum_i R_i(\boldsymbol{\theta}) \frac{dg_\Gamma}{dX_i} \tag{2.29}$$

2.4 Sensitivity Analysis

One of the most crucial steps in an optimization algorithm is the calculation of the gradients of both the objective functions as well as the constraints. In general, these functions depend on the design variables $\mathbf{x}$, as well as the response variables $\mathbf{u}$. In structural analysis, the latter correspond to displacements. These response variables are not independent but are influenced by the design variables $\mathbf{x}$, as a change in the geometry also leads to a change in displacements.

After the discretization of the geometry, the nested dependencies can generally be formulated as:

$$f_\Gamma = f_\Gamma(\mathbf{X}, \mathbf{u}(\mathbf{X})) \tag{2.30}$$

Thus, for the calculation of the gradient, we have to consider the chain rule of differentiation:

$$\frac{df_\Gamma}{d\mathbf{X}} = \frac{\partial f_\Gamma}{\partial \mathbf{X}} + \frac{\partial f_\Gamma}{\partial \mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{X}} \tag{2.31}$$

The term $\frac{d\mathbf{u}}{d\mathbf{X}}$ is called the state sensitivity. To calculate the state sensitivity, we use the state equations of the underlying problem. In structural analysis problems, this is the global stiffness equation:

$$\mathbf{r} = \mathbf{K}\mathbf{u} - \mathbf{p} = 0 \tag{2.32}$$

Taking the derivative with respect to the nodal variables $\mathbf{X}$, we get:

$$\frac{d\mathbf{r}}{d\mathbf{X}} = \frac{d\mathbf{K}}{d\mathbf{X}}\mathbf{u} + \mathbf{K}\frac{d\mathbf{u}}{d\mathbf{X}} - \frac{d\mathbf{p}}{d\mathbf{X}} = 0 \tag{2.33}$$

Solving for $\frac{d\mathbf{u}}{d\mathbf{X}}$, we obtain:

$$\frac{d\mathbf{u}}{d\mathbf{X}} = \mathbf{K}^{-1}\left(\frac{d\mathbf{p}}{d\mathbf{X}} - \frac{d\mathbf{K}}{d\mathbf{X}}\mathbf{u}\right) \tag{2.34}$$

2.4.1 Direct Method

In the direct method, the gradient of the objective function with respect to the design variables is calculated by first solving Eq. 2.34 for $\frac{d\mathbf{u}}{d\mathbf{X}}$. Then, this result is substituted into Eq. 2.31 to compute the total derivative $\frac{df_\Gamma}{d\mathbf{X}}$.

2.4.2 Adjoint Method

In the adjoint method, we first rewrite Eq. 2.31 by substituting Eq. 2.34:

$$\frac{df_\Gamma}{d\mathbf{X}} = \frac{\partial f_\Gamma}{\partial \mathbf{X}} + \frac{\partial f_\Gamma}{\partial \mathbf{u}} \mathbf{K}^{-1} \left(\frac{d\mathbf{p}}{d\mathbf{X}} - \frac{d\mathbf{K}}{d\mathbf{X}} \mathbf{u} \right) \tag{2.35}$$

We then introduce an adjoint variable $\boldsymbol{\lambda}$, which satisfies the adjoint equation:

$$\boldsymbol{\lambda}^\top \mathbf{K} = \frac{\partial f_\Gamma}{\partial \mathbf{u}}^\top \tag{2.36}$$

Once the adjoint variable $\boldsymbol{\lambda}$ is obtained, the gradient of the objective function with respect to the design variables is computed as:

$$\frac{df_\Gamma}{d\mathbf{X}} = \frac{\partial f_\Gamma}{\partial \mathbf{X}} + \boldsymbol{\lambda}^\top \left(\frac{d\mathbf{p}}{d\mathbf{X}} - \frac{d\mathbf{K}}{d\mathbf{X}} \mathbf{u} \right) \tag{2.37}$$

2.4.3 Direct vs. Adjoint

Direct sensitivity analysis is straightforward and best suited for problems with a small number of design variables due to its computational cost. In contrast, adjoint sensitivity analysis is more efficient for problems with many design variables, as it requires solving the adjoint equations only once. Adjoint methods are preferable in large-scale optimization scenarios, despite their implementation complexity, whereas direct methods are favored for simpler, smaller problems. As node-based shape optimization deals with a large number of design variables the adjoint method is used.

2.5 Optimization Loop

In the following, we will assume, that for the engineering problem, we can obtain the values for the objective function and the constraint functions, as well as their nodal gradients from a structural solver. We will treat this solver as a black box. We will use J as the notation for all response functions and G as the nodal gradients $\frac{df_\Gamma}{dX_i}$ and $\frac{dg_\Gamma}{dX_i}$. Obtaining these value, we can now bring everything together.

The optimization starts with the initial geometry. In an iterative process the current geometry is analyzed by an external finite element solver that calculates the current response values, as well as the nodal sensitivities. The gradients are mapped into the control space, where then the actual update is computed. In this work, for unconstrained optimization problems, the steepest descent method is used. For constrained problems, the gradient projection method is applied. After the update is computed in the control space, we then apply a forward mapping to obtain the new nodal coordinates of the geometry, preparing for a new step. This loop is then repeated until a convergence criterion is met, for example, the objective function meets a certain absolute value, the relative improvement between two steps is very small, or after a predefined number of iterations.

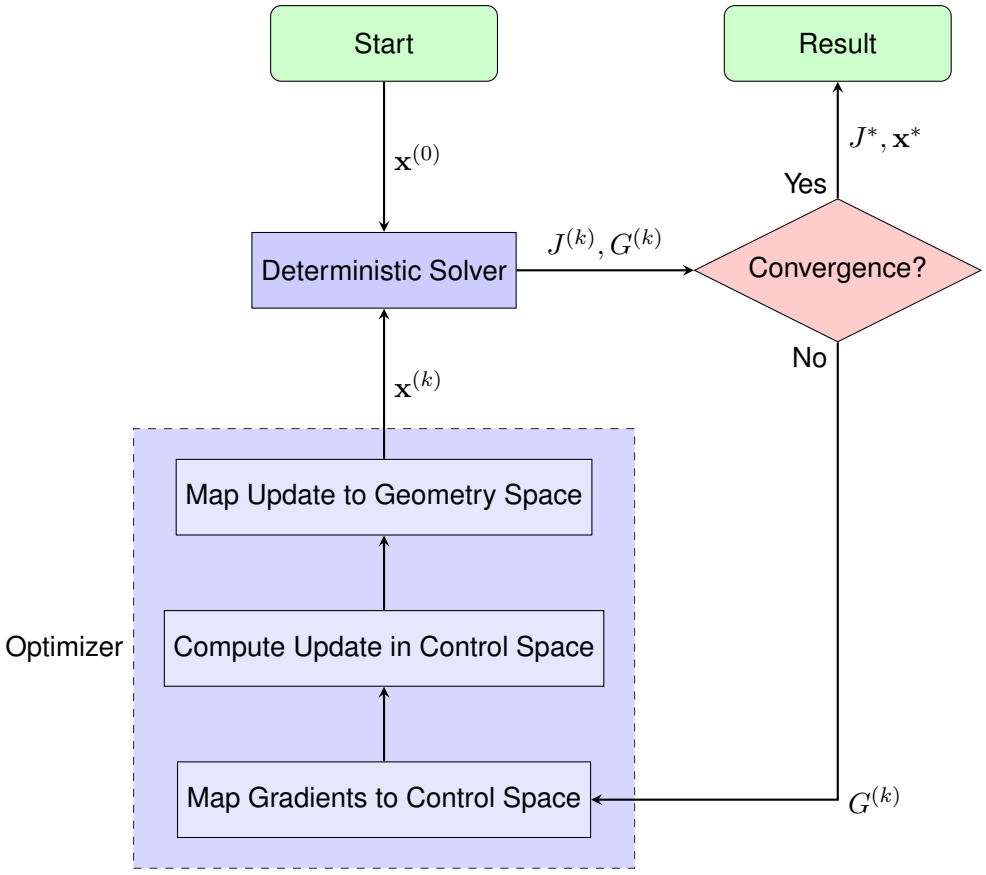


Figure 7 Flow diagram of optimization loop

Algorithm 1 shows the fundamental steps of the optimization loop in pseudo-code.

Algorithm 1 Optimization Framework using vertex morphing

1: Initialize design variables $\mathbf{x}^{(0)}$
2: Set convergence criterion
3: $k \leftarrow 0$
4: **while** convergence criterion not met **do**
5: Evaluate deterministic solver to compute J and G
6: Map gradients to control space: (Eq. 2.27)
7: Compute update in control space: (Eq. 2.24)
8: Map update to geometry space: (Eq. 2.22)
9: $k \leftarrow k + 1$
10: **end while**
11: Return optimized design $\mathbf{x}^*$

2.6 Multi-Objective Optimization

Multi-objective optimization is a branch of optimization problems in which more than one objective function is optimized simultaneously. Unlike single-objective optimization, where a single optimal solution is sought, multi-objective optimization aims to find a set of solutions that provide a trade-off among conflicting objectives. This is because improving one objective might lead to a deterioration in another.

2.6.1 Concept of Pareto Optimality

In multi-objective optimization, the concept of Pareto optimality is used to describe solutions that represent the best trade-offs among the different objectives. A solution is considered *Pareto optimal* if there is no other solution that can improve at least one objective without worsening another. Formally, given a set of objective functions $f_1, f_2, \ldots, f_k$, a solution $\mathbf{x}^*$ is Pareto optimal if there does not exist another solution $\mathbf{x}$ such that:

$$f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*) \text{ for all } i \text{ and } f_j(\mathbf{x}) < f_j(\mathbf{x}^*) \text{ for at least one } j.$$

Here, $\mathbf{x}$ is said to *dominate* $\mathbf{x}^*$ if it improves on at least one objective without harming any other.

2.6.2 Pareto Front

The *Pareto front* (or Pareto boundary) is a set of all Pareto optimal solutions in the objective space. This front provides a comprehensive view of the trade-offs between objectives and helps decision-makers choose a solution that best matches their preferences or constraints. In practice, the Pareto front is often visualized in a two-dimensional or three-dimensional plot where each point on the front represents a solution that cannot be improved in any objective without degrading another.

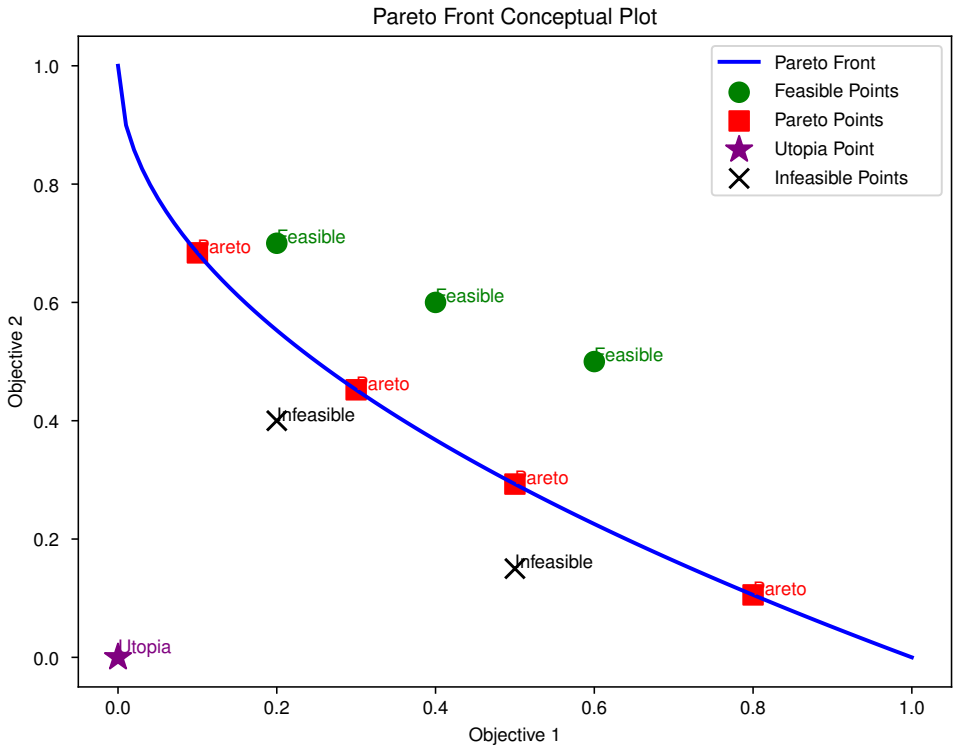


Figure 8 The Pareto front as the set of points that are not dominated by any other feasible point

In summary, multi-objective optimization provides a framework for handling problems with conflicting objectives by identifying a set of optimal solutions, known as the Pareto front, that balance these objectives. This approach helps in making informed decisions based on trade-offs and preferences.

3 Fundamentals of Uncertainty Quantification

3.1 Statistical Preliminaries

To ensure this thesis is self-contained, we briefly review the fundamental concepts, definitions, and notation of random calculus. These will be necessary to later utilize them in the robust optimization workflow. The content in this section is based on [16], [17], and [18], which are recommended for further reading.

Random Variables

Random variables form the foundation of random calculus and are classified into two categories: continuous and discrete. A random variable is denoted by X, and its realization or outcome is represented by x.

Continuous random variables are described by their cumulative distribution function (CDF) $F_X(x)$. The CDF is defined as the probability that X takes a value less than or equal to x:

$$F_X(x) = P(X \leq x). \tag{3.1}$$

The probability density function (PDF) $f_X(x)$ is obtained by differentiating the CDF:

$$f_X(x) = \frac{dF_X(x)}{dx}. \tag{3.2}$$

The probability of X falling within the interval $[a, b]$ can be computed as:

$$P(a \leq X \leq b) = \int_a^b f_X(x)\, dx. \tag{3.3}$$

For discrete random variables, the cumulative mass function (CMF) $P_X(x)$ and the probability mass function (PMF) $p_X(x)$ are defined as:

$$P_X(x) = \sum_{x_i \leq x} p_X(x_i), \tag{3.4}$$

$$p_X(x) = P(X = x_i). \tag{3.5}$$

Table 1 presents some common continuous and discrete random variables.

	Distribution	Notation	$f_X(x)$ or $p_X(x)$	Support
Continuous	Gaussian	$\mathcal{N}(\mu, \sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$(-\infty, \infty)$
	Gamma	Gamma(α, λ)	$\frac{\lambda^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\lambda x}$	$[0, \infty)$
	Beta	Beta(α, β)	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$	$[0, 1]$
	Uniform	$\mathcal{U}(\alpha, \beta)$	$\frac{1}{\beta-\alpha}$	$[\alpha, \beta]$
Discrete	Poisson	Poi(λ)	$e^{-\lambda} \frac{\lambda^x}{x!}$	$\{0, 1, 2, \ldots\}$
	Binomial	Bin(n, p)	$\binom{n}{x} p^x (1-p)^{n-x}$	$\{0, 1, 2, \ldots, n\}$
	Neg. Binomial	NB(n, p)	$\binom{x+n-1}{x} (1-p)^x p^n$	$\{0, 1, 2, \ldots\}$
	Hypergeometric	Hyp(N, K, n)	$\frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$	$\{0, 1, 2, ...$
				$..., \min(K, n)\}$

Table 1 Common probability distributions, adapted from [18].

Moments of Random Variables

Statistical moments are numerical characteristics used to describe random variables. The n-th moment $\mathbb{E}(X^n)$ of a random variable X is given by:

$$\mu_X = \mathbb{E}(X^n) = \int x^n f_X(x)\, dx. \tag{3.6}$$

The first moment, $\mu_X = \mathbb{E}(X)$, is known as the expected value or mean of X. The variance of X, denoted by σ^2, is defined as:

$$\sigma_X^2 = \mathsf{Var}(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2, \tag{3.7}$$

and measures the variability of X around its mean.

Joint Distributions

In many problems, outcomes depend on more than one random variable. Let $\mathbf{X} = (X_1, X_2, \ldots, X_N)^T$ be a vector of N random variables. The joint CDF is defined as:

$$F_{\mathbf{X}}(\mathbf{x}) = P(X_1 \leq x_1, X_2 \leq x_2, \ldots, X_N \leq x_N). \tag{3.8}$$

The joint PDF is given by:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{\partial^N}{\partial x_1 \partial x_2 \cdots \partial x_N} F_{\mathbf{X}}(\mathbf{x}). \tag{3.9}$$

Marginal PDFs can be obtained from the joint PDF by integrating. For example, the marginal PDF $f_{X_1}(x_1)$ of X_1 is:

$$f_{X_1}(x_1) = \int f_{\mathbf{X}}(x_1, x_2)\, dx_2. \tag{3.10}$$

The conditional PDF of X_1 given $X_2 = x_2$ is denoted $f_{X_1|X_2}(x_1|x_2)$ and is defined by:

$$f_{X_1|X_2}(x_1|x_2) = \frac{f_{\mathbf{X}}(x_1, x_2)}{f_{X_2}(x_2)}. \tag{3.11}$$

Random variables X_1 and X_2 are considered independent if:

$$f_{X_1|X_2}(x_1|x_2) = f_{X_1}(x_1). \tag{3.12}$$

The covariance between X_i and X_j is given by:

$$\mathsf{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_{X_i})(X_j - \mu_{X_j})], \tag{3.13}$$

which measures the linear dependence between the two variables.

The correlation coefficient is defined as:

$$\rho_{X_i,X_j} = \frac{\mathrm{Cov}(X_i, X_j)}{\sigma_{X_i}\sigma_{X_j}}, \tag{3.14}$$

and ranges from $[-1, 1]$. It is important to note that the correlation coefficient measures only linear relationships. While $\rho_{X_i,X_j} = 0$ indicates no linear correlation, it does not imply that X_i and X_j are independent.

Problem Description and Notation

In the following chapters, we examine systems that generally depend on a spatial variable $\mathbf{x}$, and an N-dimensional random vector ξ. The solution u we seek is the output of a model $\mathcal{M}$. In the context of structural optimization, this will be a response function, that is obtained through structural analysis. Since u is a function of one or more random variables, it is itself a random variable with an (unknown) PDF. The goal is to obtain information about the stochastic moments of u and, potentially, to approximate its PDF.

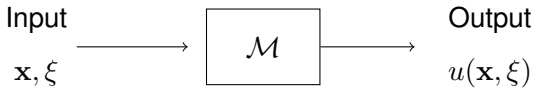


Figure 9 Problem description

3.2 Monte Carlo Method

The Monte Carlo method is one of the most popular approaches for computing expectations related to the problem described in Section 3.1. It involves the following steps:

1. Draw a finite sequence of independent and identically distributed (i.i.d.) samples $\{\xi^1, \xi^2, \ldots, \xi^{N_S}\}$ from the distribution of the random variable of interest using a pseudo-random number generator.

2. Evaluate the model for each sample, yielding $u^s = u(\xi^s)$.

3. Estimate the expectation by aggregating the results:

$$\mu_u = \mathbb{E}[u(\xi)] = \int u(\xi) f_X(\xi)\, d\xi \approx \frac{1}{N_S} \sum_{s=1}^{N_S} u^i. \tag{3.15}$$

4. Obtain the variance of the estimator:

$$\hat{\sigma}_u^2 = \mathsf{Var}(u) = \frac{1}{N_S - 1} \sum_{s=1}^{N_S} (u^i - \hat{\mu})^2, \tag{3.16}$$

   where $\hat{\mu} = \frac{1}{N_S} \sum_{i=1}^{N_S} u^i$ is the sample mean of the model evaluations. The standard deviation of the estimator is then:

$$\hat{\sigma}_u = \sqrt{\mathsf{Var}(u)}. \tag{3.17}$$

The Monte Carlo method provides a straightforward way to estimate expectations related to stochastic quantities u [18]. It only requires the ability to draw samples from the input PDF $f_X(\xi)$ and to evaluate the model at these sample points. Consequently, it imposes no specific regularity requirements on the model [19]. Additionally, because it does not require defining a grid, it is particularly well-suited for complex integration domains.

However, standard Monte Carlo estimators converge at a relatively slow rate of $\mathcal{O}(\frac{1}{\sqrt{N_S}})$ as $N_S \to \infty$. This slow convergence can be a significant drawback when high resolution is required and evaluating the deterministic model is computationally expensive. On the positive side, the convergence rate is **independent** of the dimension of ξ, making Monte Carlo estimators a popular choice for high-dimensional problems.

To improve convergence rates, several advanced techniques have been developed. These include "Quasi-Monte Carlo" methods [20], control variate methods [21], and specialized sampling techniques, which will be discussed in Section 3.3.2.

3.3 Polynomial Chaos Expansion

3.3.1 Basic Aspects and Properties

The main idea of the Generalized Polynomial Chaos Method (gPCM) is to approximate u using an expansion of the form:

$$u(\mathbf{x};\xi) \approx u^P(\mathbf{x};\xi) := \sum_{i=0}^{P} c_i(\mathbf{x})\Psi_i(\xi), \tag{3.18}$$

where Ψ_i are the expansion polynomials, and c_i are the corresponding deterministic coefficients. The number of terms $P+1$ in the expansion is determined by:

$$P+1 = \frac{(p+N)!}{p!N!}, \tag{3.19}$$

where N is the number of random variables and p is the highest polynomial degree used.

N \ p	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	6	10	15	21	28
3	4	10	20	35	56	84
4	5	15	35	70	126	252
5	6	21	56	126	252	462
6	7	28	84	252	462	924
8	9	45	165	495	1287	3003
12	13	91	455	1820	6188	18,564

Table 2 Number of terms $P+1$ in a gPCM model with respect to the highest expansion order p and the number of random variables N, computed using (3.19).

In theory, the type of polynomials used in the Polynomial Chaos Expansion can be chosen arbitrarily, and the PCM will converge in a mean-square sense. However, for certain random variables, specific polynomials can provide optimal convergence. These optimal polynomials satisfy an orthogonality condition with respect to the PDF $f_X(\xi)$ of the random variables, i.e.,

$$\int_\Gamma \Psi_i(\xi)\Psi_j(\xi)f_X(\xi)\,d\xi = \langle \Psi_i, \Psi_j \rangle = \gamma_j \delta_{ij}, \tag{3.20}$$

where δ_{ij} is the Kronecker delta and γ_j is a normalization constant.

In 1938, Wiener [22] demonstrated that for the multivariate Gaussian distribution, Hermite polynomials are optimal. For other common distributions, adapted polynomials have been identified, as summarized in the Askey table [23].

	Probability Distribution	**Askey Chaos**	**Support**
Continuous	Gaussian	Hermite-Chaos	$(-\infty, \infty)$
	Gamma	Laguerre-Chaos	$[0, \infty)$
	Beta	Jacobi-Chaos	$[a, b]$
	Uniform	Legendre-Chaos	$[a, b]$
Discrete	Poisson	Charlier-Chaos	$\{0, 1, 2, \ldots\}$
	Binomial	Krawtchouk-Chaos	$\{0, 1, 2, \ldots, N\}$
	Negative Binomial	Meixner-Chaos	$\{0, 1, 2, \ldots\}$
	Hypergeometric	Hahn-Chaos	$\{0, 1, 2, \ldots, N\}$

Table 3 Askey table, adapted from [24]. This table lists common probability distributions and their corresponding optimal polynomial chaos families.

$$
\begin{aligned}
H_0 &= 1 \\
H_1 &= \xi \\
H_2 &= \xi^2 - 1 \\
H_3 &= \xi^3 - 3\xi \\
H_4 &= \xi^4 - 6\xi^2 + 3
\end{aligned}
$$

Table 4 First five Hermite polynomials.

For any other distributions, orthogonal polynomials can still be constructed using methods such as the Gram-Schmidt process [25]. Thus, the choice of basis functions depends solely on the distribution of the random variables ξ and is independent of the function u. For independent variables ξ, the corresponding basis functions Ψ can be calculated as a product of the individual polynomial functions ψ [26]:

$$
\Psi_j(\xi) = \prod_{k=1}^{N} \psi_{i_k}(\xi_k) \quad \text{such that} \quad \sum_{k=1}^{N} i_k \leq p \quad \text{and} \quad 0 \leq j \leq N. \tag{3.21}
$$

After selecting the type of polynomial, there are various methods to calculate the coefficients $c_i(\mathbf{x})$. These methods will be discussed in Section 3.3.2. A key property of the coefficients is that the stochastic moments can be computed using a weighted sum. For the mean and

variance, we have:

$$\begin{aligned}\mathbb{E}(u) &= \left\langle \sum_{i=0}^{P} c_i(\mathbf{x})\Psi_i, 1 \right\rangle \\ &= \sum_{i=0}^{P} c_i(\mathbf{x})\langle \Psi_i, 1\rangle = u_0(\mathbf{x}).\end{aligned} \tag{3.22}$$

$$\begin{aligned}\sigma^2(u) &= \mathbb{E}(u^2) - (\mathbb{E}(u))^2 \\ &= \left\langle \sum_{i=0}^{P}\sum_{j=0}^{P} c_i(\mathbf{x})u_j(\mathbf{x})\Psi_i, \Psi_j \right\rangle - u_0^2(\mathbf{x}) \\ &= \sum_{i=0}^{P}\sum_{j=0}^{P} c_i(\mathbf{x})u_j(\mathbf{x})\langle \Psi_i, \Psi_j\rangle - u_0^2(\mathbf{x}) \\ &= \sum_{i=0}^{P} c_i^2(\mathbf{x})\langle \Psi_i, \Psi_i\rangle - u_0^2(\mathbf{x}) \\ &= \sum_{i=1}^{P} c_i^2(\mathbf{x})\langle \Psi_i, \Psi_i\rangle \\ &= \sum_{i=1}^{P} c_i^2(\mathbf{x})\gamma_i.\end{aligned} \tag{3.23}$$

3.3.2 Intrusive Methods vs. Non-Intrusive Methods

While all Polynomial Chaos (PC) methods ultimately yield a polynomial expansion as given in Eq. (3.18), their implementation and the process of calculating the expansion coefficients differ significantly. These methods can generally be categorized into two main approaches: **intrusive** and **non-intrusive**. Fig. 10 illustrates conceptual diagrams of each approach.

In **intrusive** PC methods, the polynomial expansion of the unknown variables is directly integrated into the governing equations of the model. This is achieved by replacing each unknown u with its expansion in terms of the polynomial basis and corresponding coefficients. The result is a modified set of equations where the solution is sought for the PC coefficients c_i. These methods, such as the intrusive Galerkin projection method [27], can solve for all PC coefficients in a single computation, leading to high accuracy as the stochastic system is solved directly. However, this advantage comes at the cost of complexity, as the original deterministic code must be altered. This often leads to increased implementation effort and computational expense, particularly when the modified equations result in a coupled system that affects a significant portion of the original model [24]. Despite these challenges, intrusive methods can be more accurate because they integrate the stochasticity into the core of the simulation, avoiding approximations made in non-intrusive methods [23].

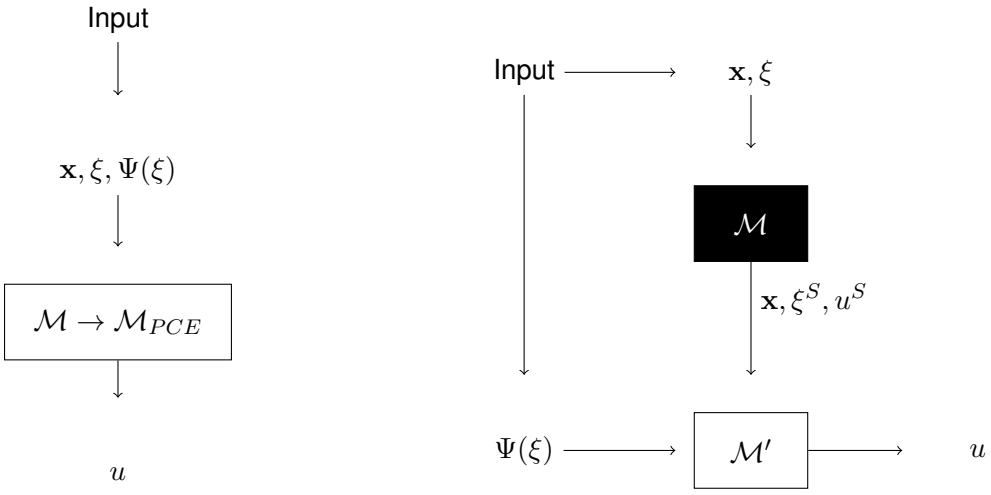
Non-intrusive PC methods, on the other hand, treat the existing deterministic model as a black box. Rather than modifying the model equations, these methods construct a surrogate model based on Eq. (3.18). The PC coefficients are determined either by **projection** or **regression**. In projection-based methods, the solution u is projected onto the polynomial basis, and the coefficients c_i are computed using a quadrature rule. This approach requires the selection of appropriate quadrature nodes and weights to evaluate the integral, ensuring that the stochastic system is well-represented.

In contrast, regression-based methods, also known as **collocation-based** PC or Point Collocation Polynomial Chaos Expansion (PCol-PCE), approximate the solution by sampling the stochastic space at N_S collocation points. For each sample, the expansion can be written as:

$$u(\mathbf{x};\xi^S) = \sum_{i=0}^{P} c_i(\mathbf{x})\Psi_i(\xi^S), \tag{3.24}$$

which results in a system of N_S equations with $P+1$ unknowns. This overdetermined system is solved using a regression technique, where the goal is to minimize the residuals between the system evaluations and the polynomial approximation.

Non-intrusive methods are often easier to implement because they do not require modification of the original code and can leverage existing deterministic solvers. However, they may be less accurate or require more computational effort due to the need for multiple evaluations of the deterministic model, especially when using regression or high-order quadrature methods.



(a) Intrusive PCE

(b) Non-intrusive PCE

Figure 10 Comparison of PCE approaches

Sampling Techniques

The number of samples and the method by which they are drawn for Monte Carlo methods (see Section 3.2) and PCol-PCE can vary significantly. Samples can be drawn either randomly (or pseudo-randomly) from the PDF of the random variable or selected deterministically. Random sampling involves generating samples purely based on probability, while deterministic sampling selects points in a structured manner to ensure better coverage of the PDF. Although deterministic sampling can represent higher-order moments more accurately, it becomes computationally expensive when attempting to represent these moments with high precision.

Several techniques combine elements of both random and deterministic sampling to achieve more efficient sampling. Notable examples include Latin Hypercube Sampling (LHS) [28], Hammersley Sampling (HS) [29], and Sobol sequences [30]. Unlike purely random sampling, both LHS and HS divide the cumulative distribution function (CDF) of the random variable into equal segments and then generate a sample within each segment. This process ensures that the CDF is sampled more uniformly, resulting in a more complete representation of the distribution [31]. Sobol sequences, in contrast, are a sequence of points within the uniform unit hypercube where each new point is chosen based on the position of previously sampled points. This approach ensures an even distribution of points without clustering or gaps. The samples generated by Sobol sequences can then be transformed to match other

random distributions [32]. It should be noted that the convergence behavior of Monte Carlo methods described in Section 3.2 holds strictly for independent and identically distributed (iid) random samples.

Fig. 11 provides a visual representation of standard normal CDFs using 10 two-dimensional samples generated by different sampling techniques. In Fig. 11a, purely random sampling leads to a disproportionate number of samples in the upper half of the CDF. Specifically, when dividing the CDF into five equal intervals, five samples are concentrated in the interval $[0.8, 1]$, while only one sample for x_1 and zero samples for x_2 were drawn from the interval $[0, 0.2]$. This uneven distribution results in an incomplete representation of the PDF, but it should be noted that this is the result of a single random sampling instance and is not generally repeatable.

In contrast, Fig. s 11b and 11c show LHS and HS samples, which ensure more comprehensive coverage of the CDF. LHS ensures that each segment of the CDF is sampled once for each variable, while HS uses a more deterministic approach where the first variable's samples are endpoints of each segment, and subsequent variables are sampled based on a van der Corput sequence [33]. The Sobol sequence, as shown in Fig. 11d, provides an even more homogeneous coverage of the CDF by avoiding clusters and gaps.

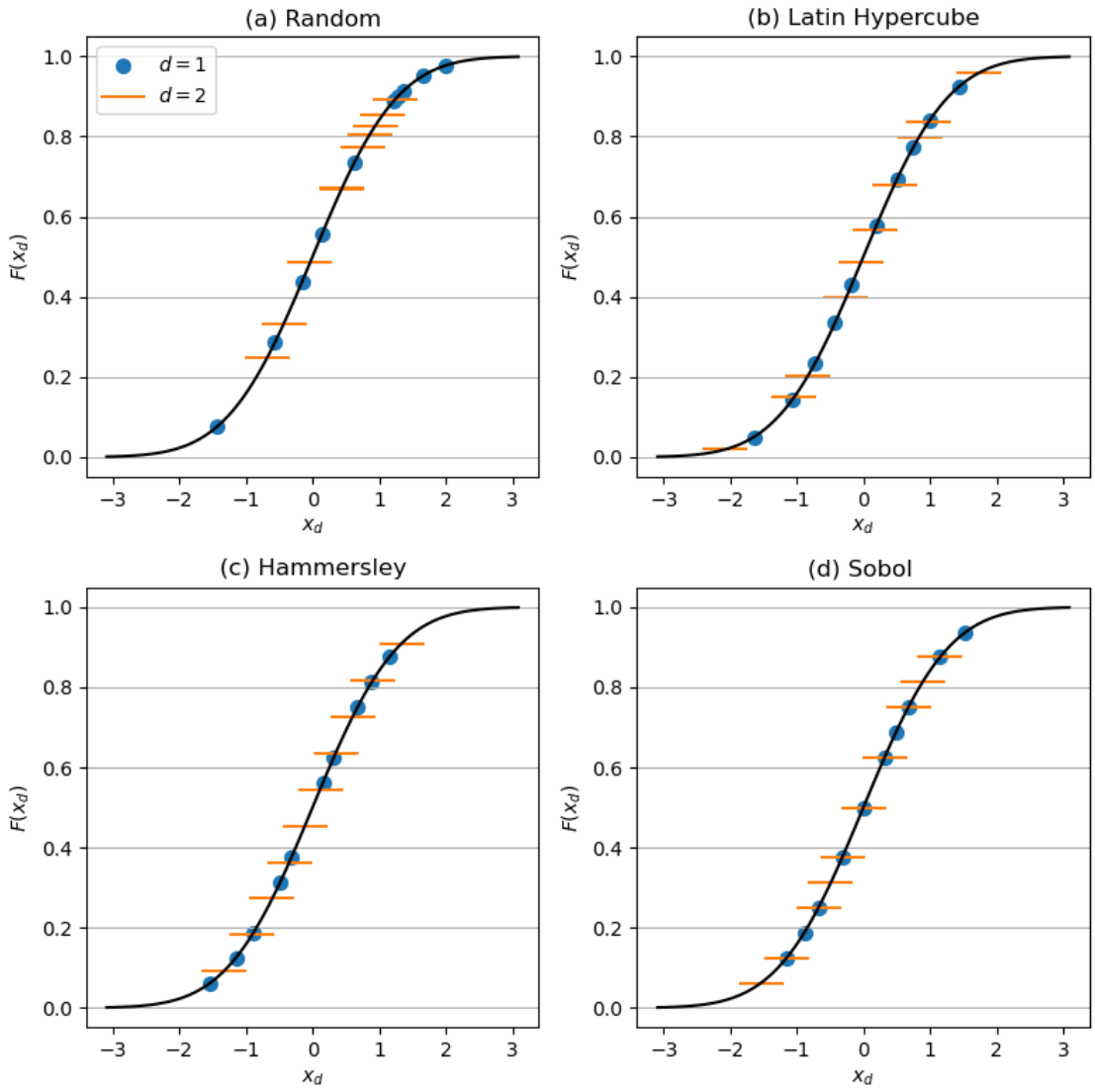


Figure 11 Two-dimensional sampling with $N_S = 10$ of a standard normal distribution for four different sampling techniques, adapted from [31].

Fig. 12 presents the bivariate normal samples for each of the sampling techniques. While both random and LHS samples in Fig. s 12a and 12b appear random, the deterministic nature of HS and Sobol sequences becomes evident in Fig. s 12c and 12d, where the samples are more evenly spaced.

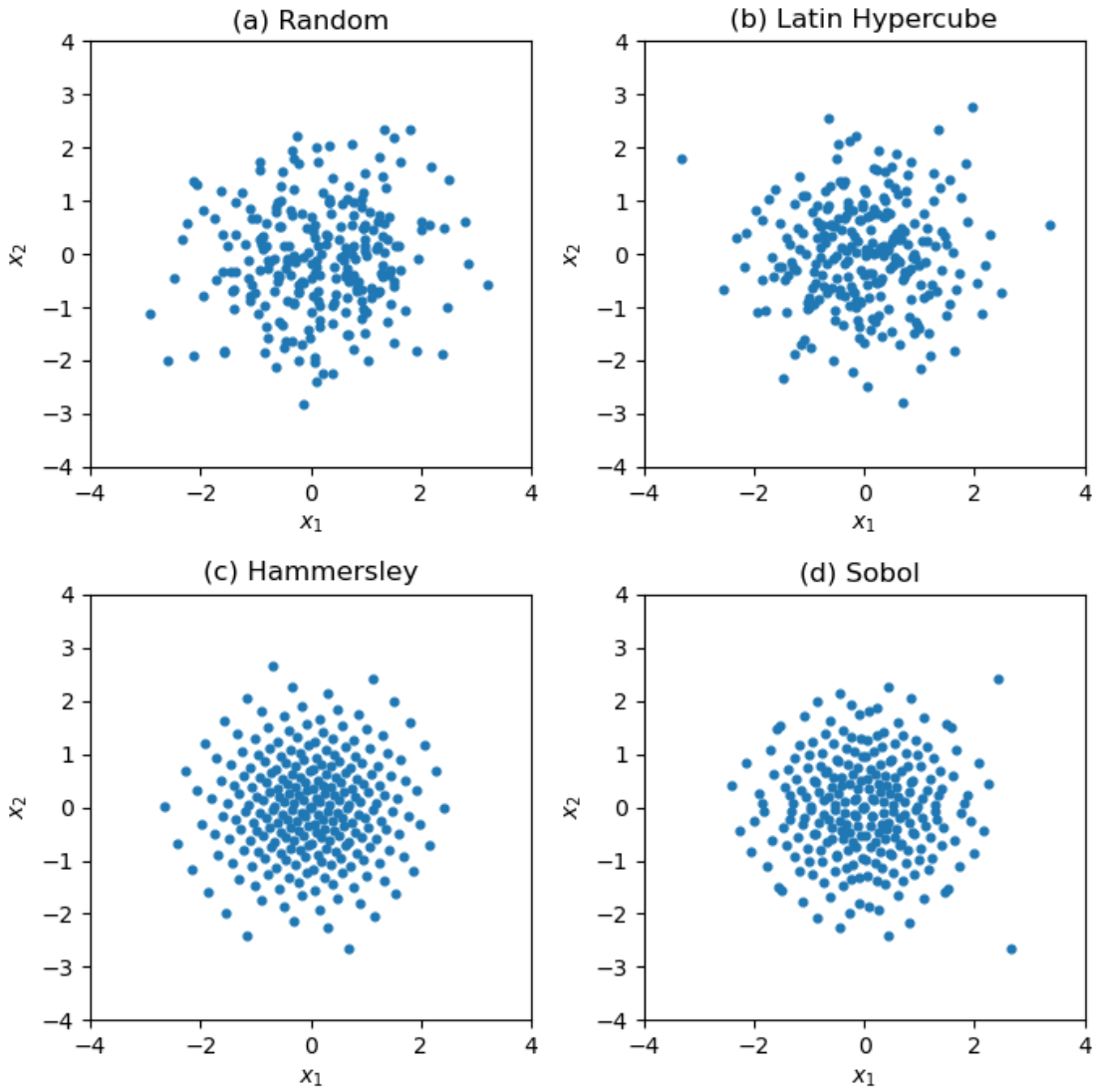


Figure 12 200 samples of a bivariate standard Gaussian distribution using different sampling techniques, adapted from [31].

Sampling for PCoI-PCE

For PCoI-PCE, Hosder et al. [34] demonstrated that Random Sampling (RS), LHS, and HS all performed similarly in terms of accuracy and computational efficiency[1]. However, LHS and HS showed a smoother and more monotonic convergence of statistical moments, making them more reliable for certain applications.

Another key consideration in PCE, as discussed by Hosder et al. [34], is the relationship between the number of samples N_S and the number of terms in the polynomial expansion $P + 1$. This relationship is often expressed through the ratio n_p, defined as:

[1] Sobol sequences were not investigated in this study.

$$n_p = \frac{N_S}{P+1} \tag{3.25}$$

Although this ratio can vary depending on the problem, a value of $n_p = 2$ often provides the best balance between accuracy and computational cost. This serves as a useful rule of thumb. When n_p approaches 1, the risk of "overfitting" increases, where the polynomial expansion too closely follows the sample points. In cases where $n_p = 1$, the polynomial expansion interpolates exactly between the sample points, leading to poor predictions for points not in the sample set. Conversely, if n_p is much larger than 2, increasing the polynomial degree improves the approximation of u and accelerates the convergence of the mean [35].

3.4 Monte Carlo Methods vs Non-intrusive Polynomial Chaos Expansion

MC methods and non-intrusive PCE both aim to quantify uncertainty, but they differ significantly in performance. Monte Carlo methods tend to be faster for problems with very high dimensionality (many random variables) or when the system exhibits highly non-linear or non-smooth behavior [36]. In such cases, PCE would require a high polynomial degree, increasing its computational cost. MC methods when low-accuracy estimates are also acceptable for the benefit of computational speed, as they can generate results incrementally with more samples improving accuracy over time. Non-intrusive PCE is generally faster when the system is smooth and well-behaved, and when the dimensionality is moderate. In these cases, PCE can achieve accurate results with far fewer samples than MC, thanks to its use of polynomial approximation, which captures the relationship between input uncertainties and outputs efficiently [37]. For problems where smoothness and moderate dimensions are present, PCE can drastically reduce the number of model evaluations compared to MC methods, making it faster for high-accuracy results.

Later on, we will use the non-intrusive Polynomial Chaos Expansion as our main method for Uncertainty Quantification and rely on a Monte Carlo approach to verify results.

4 Robust Shape Optimization

4.1 Deterministic Design

Deterministic design refers to the process of engineering or optimizing systems based on fixed, known parameters without accounting for variability or uncertainty. In this approach, every parameter in the model is considered to have a specific value, leading to a unique solution. We can store these model parameters in the parameter vector $\mathbf{P_0}$. The primary advantage of deterministic design is its simplicity and clarity. Engineers and designers can work with precise numbers, leading to specific, predictable outcomes.

We can write the optimization problem as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{f(\mathbf{x}, \mathbf{P_0})\} \tag{4.1}$$

4.2 Robust Design

Robust design aims to create systems that are resilient to variations and uncertainties in parameters. Unlike deterministic design, robust design acknowledges that real-world systems are subject to variability and seeks to minimize the impact of this variability on system performance. The goal is to ensure that the system performs reliably under a wide range of conditions.

Robust design can be divided into two main categories: non-probabilistic and probabilistic formulations.

4.2.1 Non-probabilistic Robust Design Formulations

Non-probabilistic robust design formulations do not rely on probability distributions to model uncertainties. Instead, they use other mathematical approaches to account for variability.

Multi-Optimization

Multi-optimization involves optimizing multiple objectives simultaneously. Different parameter vectors $\mathbf{P_j}$ are generated to cover the variability of the parameters. The objective function is then simply a weighted average:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{ \sum_{j=1}^{N_m} w_j f(\mathbf{x}, \mathbf{P_j}) \} \tag{4.2}$$

Worst Case Approach

The worst-case approach focuses on ensuring the system performs acceptably, even under the most adverse conditions. Designers identify the worst-case scenario for each uncertain parameter and optimize the system to function under these extreme conditions. This approach provides a high level of reliability but can be overly conservative, leading to excessive costs or reduced performance under normal conditions.

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{ \max_{P_j} \{ f(\mathbf{x}, \mathbf{P_j}) \} \} \tag{4.3}$$

Interval Analysis

Interval analysis involves defining uncertain parameters as intervals rather than precise values. The system is then designed to perform satisfactorily for any value within these intervals. This method is particularly useful when the exact distribution of uncertainties is unknown, allowing for a more flexible and encompassing design. Interval analysis ensures that the system remains functional across the entire range of possible parameter values.

4.2.2 Probabilistic Robust Design Formulations

In the probabilistic approach, the uncertain parameters are modeled as random variables. In this case they are defined either with their statistical moments or their PDF. Instead of a fixed parameter vector $\mathbf{P_0}$, they are compromised of d_ξ independent random variables $\xi \in \mathbb{R}^{d_m}$. Under uncertainty,the quantity of interest (QoI) J is not any more deterministic and becomes a random variable. The optimization problem is thus shifted towards a statistic of the QoI.

Mean Optimization

In mean optimization, the goal is to obtain the design with the lowest expected value. This approach is useful when the primary concern is the overall performance rather than the variability of outcomes.

The optimization problem can be written as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{\mu_f(\mathbf{x})\} \tag{4.4}$$

Standard Deviation Approximation

Standard deviation optimization aims to minimize the variability in system performance due to uncertainties. By reducing the standard deviation, designers can make the system more consistent and reliable. This approach is particularly valuable when the consistency of performance is crucial, even if it comes at the expense of slightly lower mean performance.

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{\sigma_f(\mathbf{x})\} \tag{4.5}$$

In practice, this approach is seldom used. Instead, a combination with the minimization of the mean is applied

Combined Mean and Standard Deviation Optimization

This approach involves optimizing both the mean performance and the standard deviation simultaneously. The goal is to achieve a balance between high average performance and low variability. This method provides a comprehensive solution, ensuring that the system is both effective and reliable.

The optimization problem turns into a multi-objective optimization. One seeks to find different designs that are located at a pareto front to compare them.

A trivial, but often-used approach is to do a weighted optimization:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{w_\mu \mu_f(\mathbf{x}) + w_\sigma \sigma_f(\mathbf{x})\} \tag{4.6}$$

By varying the weights, one can then generate different designs to choose from. This generates an approximation of the pareto front.

Quantile Optimization

Quantile optimization targets specific points in the distribution of performance metrics, such as the median or specific percentiles. For example, optimizing the 90th percentile ensures that 90% of the outcomes meet a certain performance level. This approach is useful when there are specific performance targets that must be met with high confidence.

The difference between the different formulations can be seen in Fig. 13.

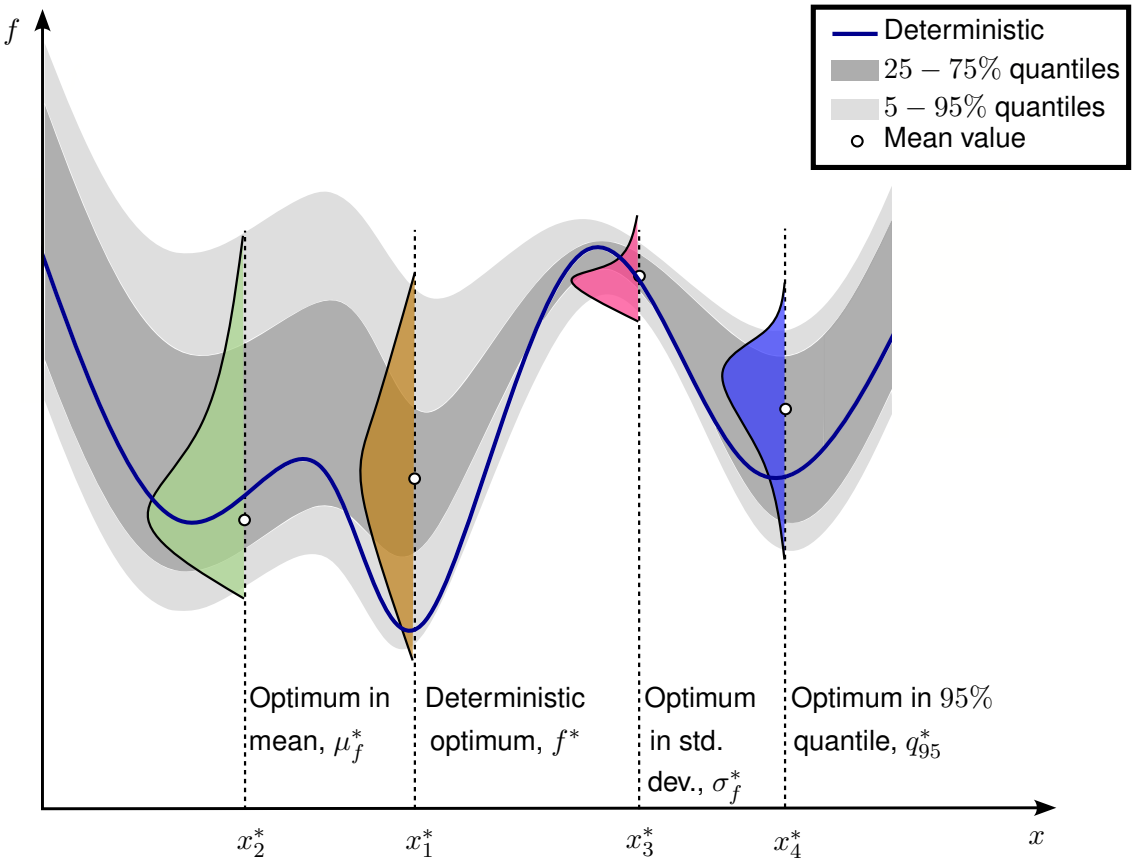
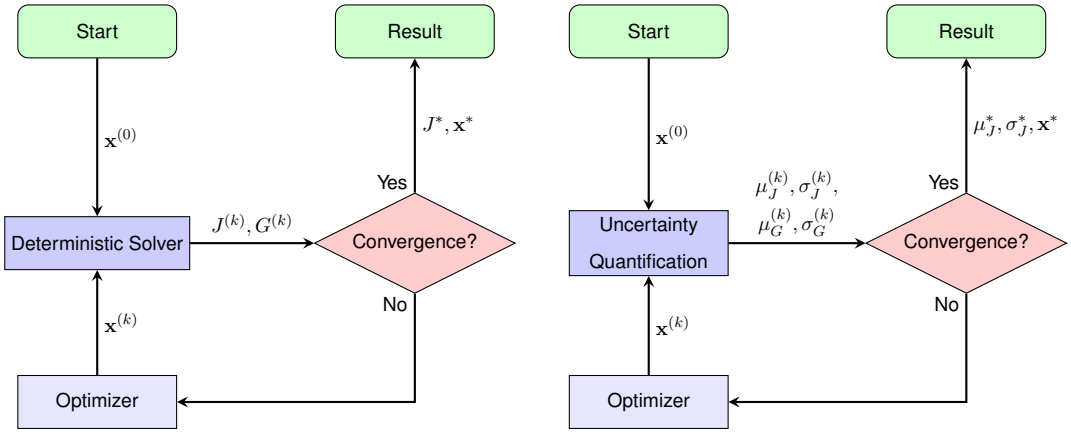


Figure 13 Comparison of different robust optimization formulations, adapted from [3]

Optimization under Uncertain Constraints

In most robust optimization approaches the above techniques are applied to the objective function. However, also the constraints are often influenced by uncertain parameters. Instead of ensuring a certain deterministic constraint, we can enforce that either the mean, a combination of mean and standard deviation, or a certain quantile of probability is fulfilled.

In this work, the *Combined Mean and Standard Deviation Optimization* for the objective function, as well as for the constraints will be applied. Fig. 15 shows the difference between the deterministic and robust optimization loop.



(a) Deterministic Optimization (b) Robust Optimization

Figure 14 Comparison of the deterministic and the combined mean and standard deviation optimization

4.3 Proposed Framework

For the step "Uncertainty Quantification" in Fig. 15, various methods can be applied. When focusing on obtaining a robust design by Combined Mean and Standard Deviation Optimization, we can apply the two methods introduced in Chapter 3, i.e., Monte Carlo estimation and the non-intrusive Polynomial Chaos expansion. Both methods aim to obtain the same relevant quantities for the optimization:

1. The mean value of the stochastic response function: $\mu_J = \mathbb{E}(J)$
2. The standard deviation of the stochastic response function: $\sigma_J = Std(J)$
3. The mean value of the gradients of the stochastic response function: $\mu_G = \mathbb{E}(G)$
4. The standard deviation of the gradients of the stochastic response function: $\sigma_G = Std(G)$

4.3.1 Robust Optimization with Monte Carlo Estimation

Following the steps described in Section 3.2, we first need to draw samples from the input vector ξ^S. The deterministic solver then evaluates the response function J, as well as the nodal gradients $G(\mathbf{X})$ for each sample.

The moments of the stochastic response function J can be calculated using Monte Carlo estimation as:

$$\mu_J, = \frac{1}{N_S} \sum_{s=1}^{N_S} J^s \tag{4.7}$$

$$\sigma_J = \sqrt{\frac{1}{N_S} \sum_{s=1}^{N_S} \left[J^s - \mu_J\right]^2} \tag{4.8}$$

As the quantity of interest J is independent of the spatial position, we can calculate the gradient of the mean as the mean of the gradients:

$$\mu_G = \frac{d\mu_J}{d\mathbf{x}} = \frac{1}{N_S} \sum_{s=1}^{N_S} G^s \tag{4.9}$$

The gradient of the standard deviation is given by:

$$\sigma_G = \frac{1}{N_S \sigma_J} \sum_{s=1}^{N_S} \left(J^s - \mu_J\right) \left(G^s - \mu_G\right) \tag{4.10}$$

4.3.2 Robust Optimization with Polynomial Chaos Expansion

To use polynomial chaos expansion (PCE) for the determination of the statistics of the response and the gradients, we approximate the response function J as:

$$J(\mathbf{x}, \boldsymbol{\xi}) = \sum_{i=0}^{P} a_i \Psi_i(\boldsymbol{\xi}) \tag{4.11}$$

The nodal gradients G can also be expressed as a PCE:

$$G = \nabla J = \sum_{i=0}^{P} \nabla a_i \Psi_i(\boldsymbol{\xi}) = \sum_{i=0}^{P} b_i(\mathbf{x}) \Psi_i(\boldsymbol{\xi}) \tag{4.12}$$

The response coefficients a_i are determined using the point-collocation PCE approach by evaluating the response function for N_S samples J^i and solving a regression problem.

Similarly, the coefficients b_i of the gradient can be calculated by determining the gradients $G^i(\mathbf{x})$ at each node for each stochastic sample. The coefficients b_i are then also determined via least-squares regression.

The statistics of the response and gradient can be calculated as:

$$\mu_J = \mathbb{E}(J) = a_0 \tag{4.13}$$

$$\sigma_J^2 = \mathsf{Var}(J) = \sum_{i=1}^{P} a_i^2 \langle \Psi_i^2 \rangle \tag{4.14}$$

$$\mu_G = \mathbb{E}(G) = b_0 \tag{4.15}$$

$$\sigma_G^2 = \mathsf{Var}(G) = \sum_{i=1}^{P} b_i^2 \langle \Psi_i^2 \rangle \tag{4.16}$$

The standard deviation of the gradient can be expressed as:

$$\sigma_G = \frac{1}{\sigma_J} \sum_{i=1}^{P} a_i b_i \langle \Psi_i^2 \rangle \tag{4.17}$$

Using the above equations, we now have all we need for the robust optimization loop. The overall loop is similar to the deterministic loop, with the main difference that instead of directly using the deterministic response function value J and the deterministic gradients G from the deterministic solver, we use the deterministic solver to evaluate samples. The samples are then used to build a PCE approximation, from which we then determine the mean and standard deviation. These are then fed back to the optimization steps to generate a new shape $\mathbf{x}^k$. The loop stops after a predefined convergence criterion is met.

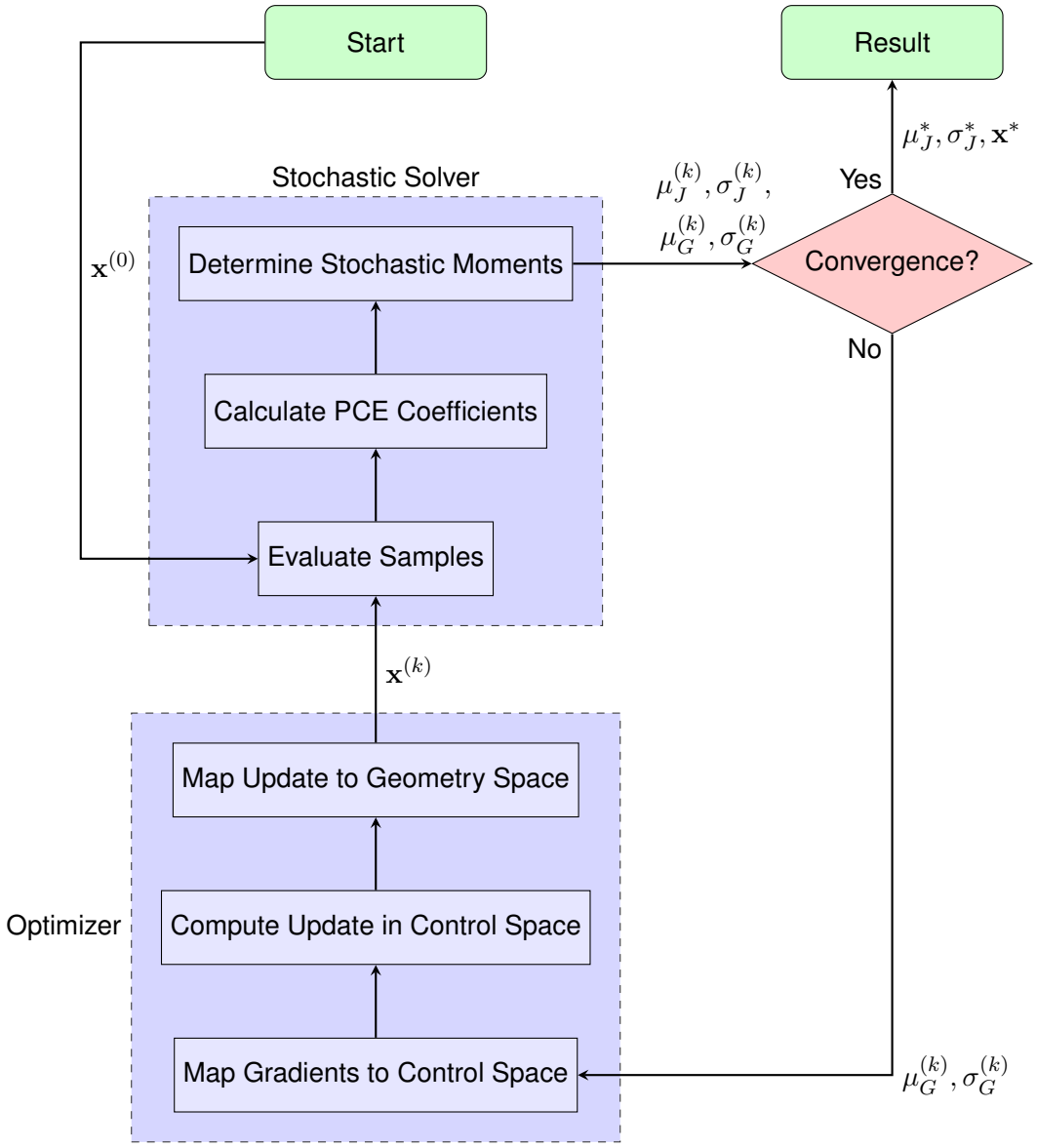


Figure 15 Final Optimisation Framework using PCE

Additionally, Algorithm 2 shows the workflow of the mean and standard deviation optimization using PCE in pseudo-code.

Algorithm 2 Robust Optimization Framework using PCE and vertex morphing

Initialize design variables $\mathbf{x}^{(0)}$
Set convergence criterion
$k \leftarrow 0$
while convergence criterion not met **do**
    Draw N_S samples of the random input $\boldsymbol{\xi}^S$
    for each sample $s = 1, \ldots, N_S$ **do**
        Evaluate deterministic solver to compute J^s and G^s
    end for
    Compute PCE coefficients a_i for response function J using regression
    Compute PCE coefficients b_i for gradients G using regression
    Calculate mean and standard deviation:
    $\mu_J^{(k)} = a_0$, $\sigma_J^{(k)} = \sqrt{\sum_{i=1}^{P} a_i^2 \langle \Psi_i^2 \rangle}$
    $\mu_G^{(k)} = b_0$, $\sigma_G^{(k)} = \frac{1}{\sigma_J^{(k)}} \sum_{i=1}^{P} a_i b_i \langle \Psi_i^2 \rangle$
    Calculate weighted objective: $J_{\text{robust}}^{(k)} = w_\mu \mu_J^{(k)} + w_\sigma \sigma_J^{(k)}$
    Calculate weighted gradients: $G_{\text{robust}}^{(k)} = w_\mu \mu_G^{(k)} + w_\sigma \sigma_G^{(k)}$
    Map gradients to control space: (Eq. 2.27)
    Compute update in control space: (Eq. 2.24)
    Map update to geometry space: (Eq. 2.22)
    $k \leftarrow k + 1$
end while
Return optimized design $\mathbf{x}^*$

5 Results

The previously presented framework structure was implemented in the open-source software framework *Kratos Multiphysics* [38]. To test the effectiveness of finding robust designs, two benchmark examples will be explored. Both of them were adapted from existing deterministic models from the Examples repository of Kratos on GitHub.

5.1 Randomly Loaded Arch Structure

5.1.1 Problem Description

The first example is of academic nature and consists of an arch structure which is supported and loaded with a line load (see Fig. 16).

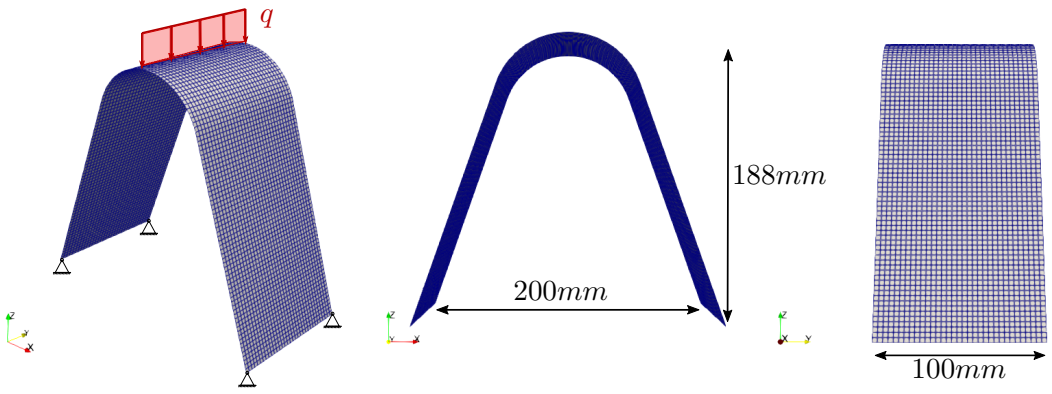


Figure 16 Dimensions of arch structure with line load

The goal of the design process is to find a structure that is of maximal stiffness. To achieve this, we want to minimize the total strain energy.

The arch structure is discretized with 4998 nodes that form 4818 elements. The material properties are displayed in Table 5

Youngs modulus	40000 N/mm
Poisson ratio	0.3
Thickness	0.8 mm
Line load	3.4N/mm

Table 5 Material properties of arch structure

Optimization Formulation

The shape optimization problem can be formulated as:

$$\begin{aligned} &\min_{\mathbf{x}} \quad U(\mathbf{x}) = \frac{1}{2}\mathbf{u}^T\mathbf{K}(\mathbf{x})\mathbf{u} \\ &\text{subject to} \\ &\mathbf{K}(\mathbf{x})\mathbf{u} = \mathbf{f}, \\ &\mathbf{u} = \mathbf{u}_0 \quad \text{on } \Gamma_u, \end{aligned} \tag{5.1}$$

For both the deterministic as well as the robust optimization, a linear filter function with a filter radius of $r = 15mm$ is used. The steepest descent method with a constant step size of $\alpha = 0.5$ is applied. The calculation of the strain energy response and its gradients takes approximately $3.8s$ with the *Kratos Multiphysics StructuralAnalysisApplication* on an *Intel(R) Core(TM) i5-12450H* with 2.00 GHz.

Modeling of Uncertainties

In the deterministic approach, we assume that the load is always acting perfectly vertically on the top nodes. However, in a real-world use case, we might not be certain that this is the case. In this academic example, we model that the load is constant along the line but that the load angle might be variable. In the following it is assumed that this angle varies uniformly both around the x-axis and the y-axis between -45 and +45 degrees. The underlying random variables are thus:

$$\begin{aligned} \xi_1 &\sim \mathcal{U}(-45, 45) \\ \xi_2 &\sim \mathcal{U}(-45, 45) \end{aligned} \tag{5.2}$$

More sophisticated ways of representing an unknown load could be chosen. However, this simplified approach is sufficient to demonstrate the differences between the deterministic and the robust designs.

5.1.2 Deterministic Optimization

Fig. 17 shows the convergence history of the deterministic optimization. After 40 steps, the strain energy is reduced by 99.8%. We can see that the relative improvement decays with increasing iterations.

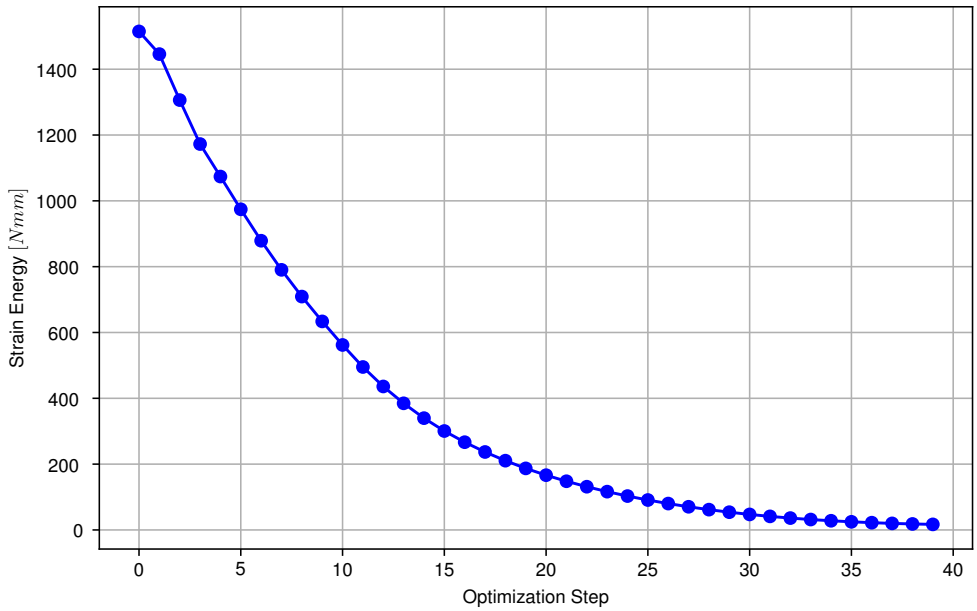
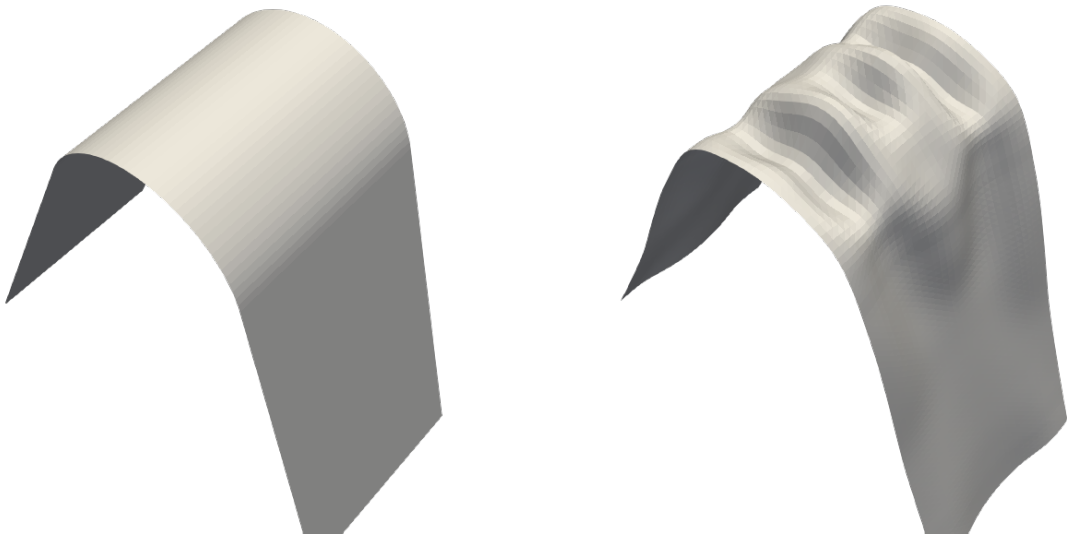


Figure 17 Convergence history of the deterministic optimization

Fig. 18 shows the initial and the final geometry after 40 Iterations. One can see that a folding pattern on top of the structure has emerged. The sides of the structure are now curved a bit inside, in contrast to the perfectly straight initial geometry. Judging by the optics, the mesh distortion stays within a reasonable range.



(a) Initial Geometry

(b) Geometry after 40 Optimization steps

Figure 18 Initial and final geometry with deterministic optimization

5.1.3 Uncertainty Quantification

In order to study the accuracy of the Uncertainty Quantification methods we want to use for the optimization, we calculate the statistical quantities of the strain energy of the initial geometry. As reference statistics, 10000 Monte Carlo samples are evaluated. Based on that, we can compare the estimations we get using Polynomial Chaos expansion and choose an appropriate sample size for our optimization.

In this case, the number of random variables is 2. With Eq. 3.19 to calculate the number of polynomial terms and the rule of thumb of $N_S = 2P$ samples, we can calculate the number of samples needed for each polynomial order, displayed in Table 6.

p	1	2	3	4
N_S	6	12	20	30

Table 6 Number of samples used for different PCE order

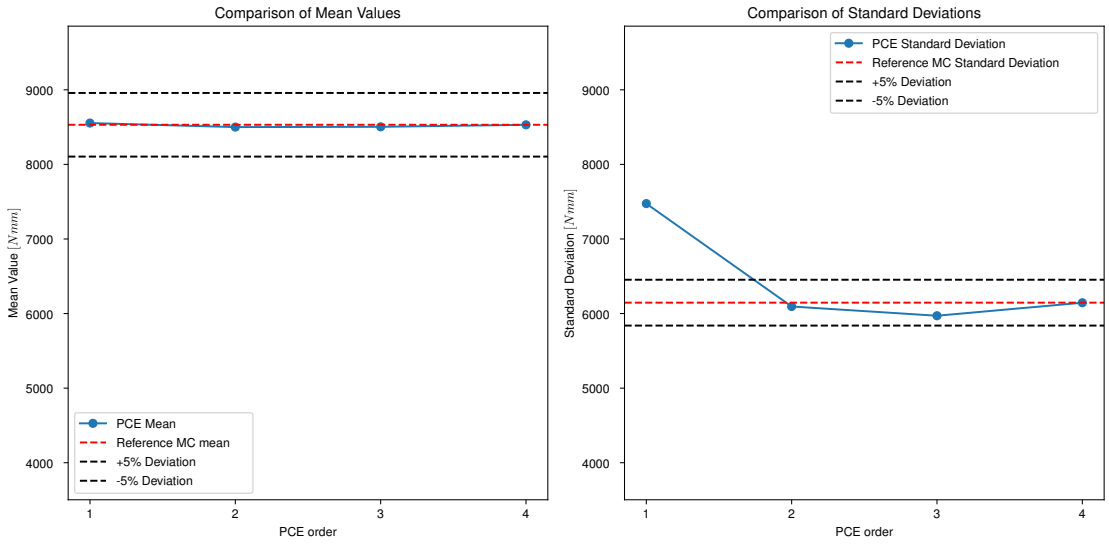


Figure 19 Approximation quality for different PCE order compared to a MC estimation using 10000 random samples

Fig. 19 shows how well PCE is able to approximate the reference solution. In 19a we can see that already a polynomial order of $p = 1$, calculated with 6 samples, is sufficient to approximate the mean value well within a 5% accuracy. However, Fig. 19b shows that with the linear approximation, the standard deviation is drastically underestimated. Since standard deviation relies on the square of deviations, which is a non-linear concept, a linear polynomial expansion (which only models linear relationships) cannot adequately capture the necessary information about the spread or variability of the data. This makes it unsuitable for calculating the standard deviation accurately. Yet, by increasing the polynomial degree to 2 and the sample size to 12, we can see that the standard deviation is now captured well within 5% accuracy. Here it should be noted that this 5% accuracy criterion of the objective function does not necessarily mean a 5% accuracy of each of the nodal gradients, nor does it guarantee a 5% accuracy for the objective function of later steps. Subsection 5.1.5 will show how different sample sizes perform.

5.1.4 Robust Optimization

The robust optimization is carried out four times with different weights for mean and standard deviation. For this section, the PCE workflow described in Section 4.3 is used with a polynomial degree of $p = 2$ and number of samples $N_s = 12$. We will investigate whether this low number of samples is enough to obtain more robust designs that perform well under uncertainty. As a sampling strategy, LHS is employed. To compare the different results, each optimization is stopped after 40 steps. After the final iteration, the geometry is then loaded with 1000 Quasi-Monte Carlo samples to be able to get the statistical differences.

Fig. 20 shows the resulting pareto front of the designs. As expected, we can see that all of the robust designs are dominating the deterministic. This is not surprising, as the deterministic

optimization does not have any information about the perturbed angles and just finds the optimum for the perfectly perpendicular line force.

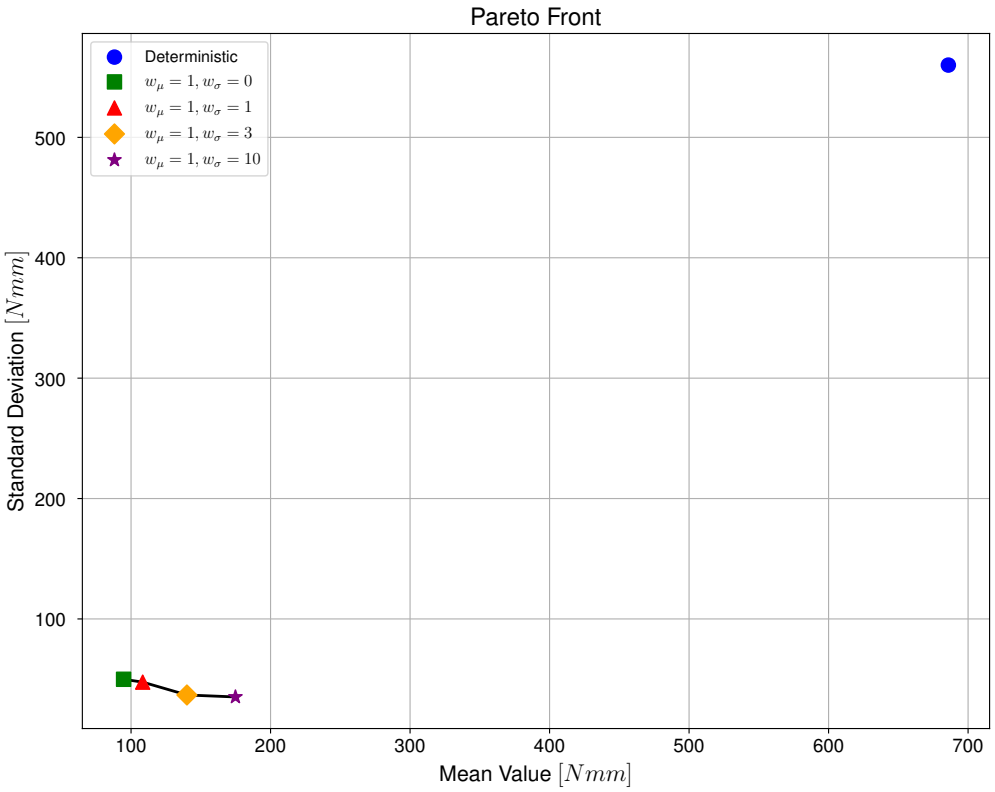


Figure 20 Approximation of Pareto front after 40 iterations

We can also see, that the four robust designs are performing very similarly. The final statistical moments are additionally displayed in Table 7.

Optimization	$\mu[Nmm]$	$\sigma[Nmm]$
Deterministic	685.9	560.0
$w_\mu = 1, w_\sigma = 0$	97.2	51.8
$w_\mu = 1, w_\sigma = 1$	108.3	47.5
$w_\mu = 1, w_\sigma = 3$	155.6	36.7
$w_\mu = 1, w_\sigma = 10$	174.8	35.7

Table 7 Statistical Moments for deterministic and robust optimization with different weights

To further analyze the results, Fig. 21 shows a violin plot of the final geometries. In this plot, the final dataset is used to obtain an estimate of the density distribution and to gain an insight about the spread and central densities of the estimated distribution. We can see that the deterministic optimization has a wider spread than the robust results. Again, the 4 robust designs perform quite similarly, with the designs with a stronger weighing of the standard deviation having a higher mean, but slightly lower spread.

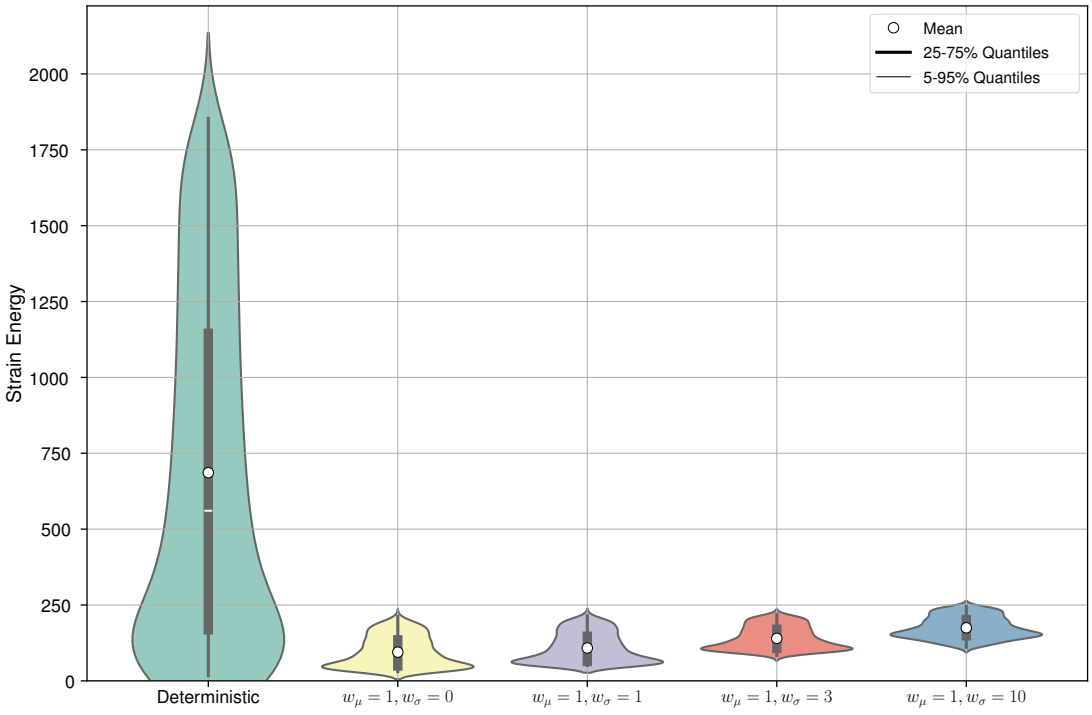
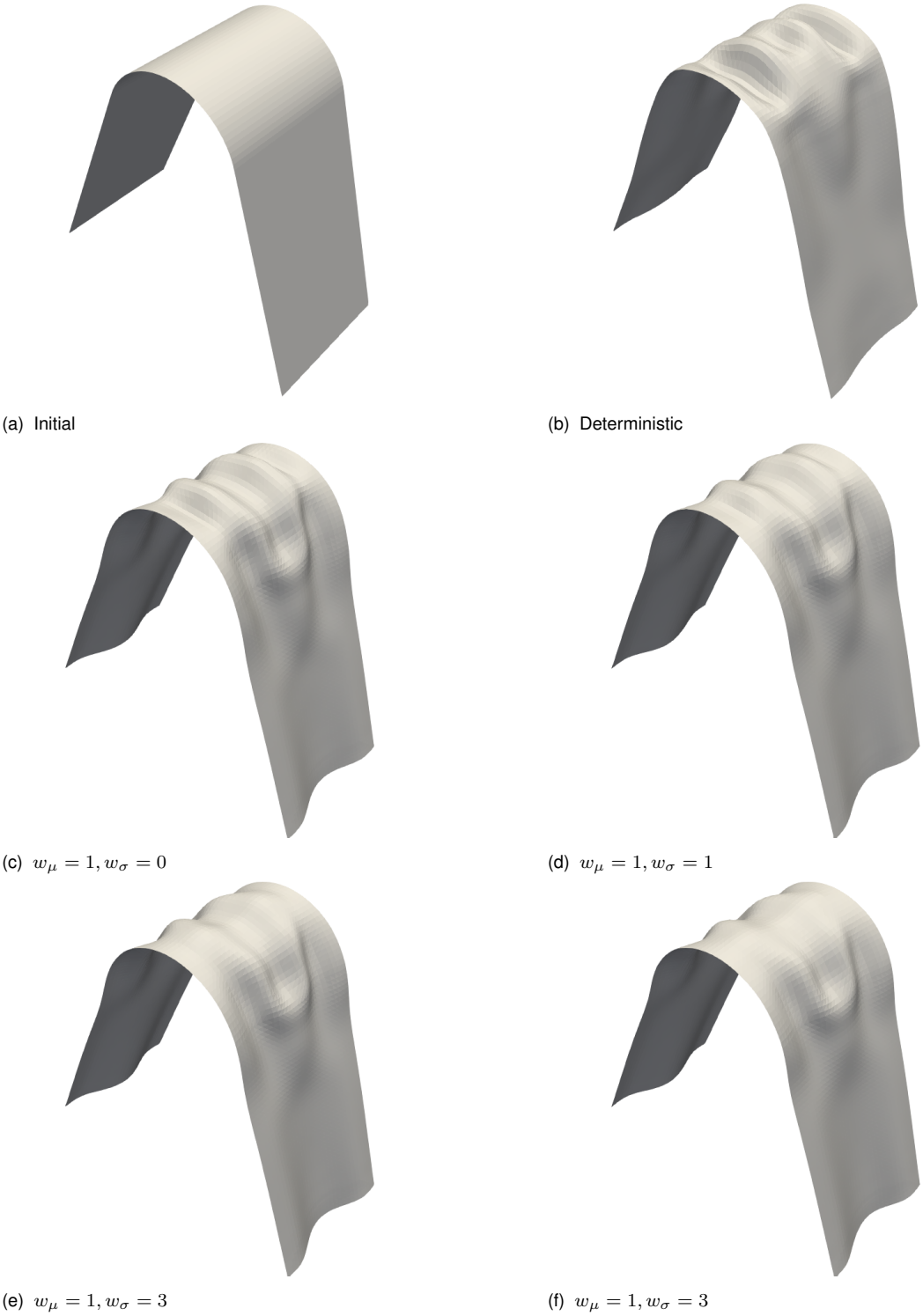


Figure 21 Violin plot of strain energy

The statistical analysis shows that the designs emerging from the robust optimization perform much better under uncertainty than the deterministically optimal design. It should be noted again that this is due to the nature of the difference of information of the load angles. Nonetheless, it shows that the proposed optimization framework is able to iteratively generate these statistically superior designs.

Shape

It is also interesting to compare the final geometries to each other. We can see that the robust arch structures show a different structure of folds than the deterministic geometry. Within the four robust results, we can see the same folding pattern. However, the height of the indentations decreases with increasing weight of the standard deviation.



(a) Initial

(b) Deterministic

(c) $w_\mu = 1, w_\sigma = 0$

(d) $w_\mu = 1, w_\sigma = 1$

(e) $w_\mu = 1, w_\sigma = 3$

(f) $w_\mu = 1, w_\sigma = 3$

Figure 22 Initial and optimized arch geometries

5.1.5 Study of Accuracy

To study the accuracy of the initial robust optimization ($p = 2$,$N_S = 12$), we can repeat the optimization and vary the sample size, as well as the polynomial degree used for the PCE surrogate.

Mean optimization

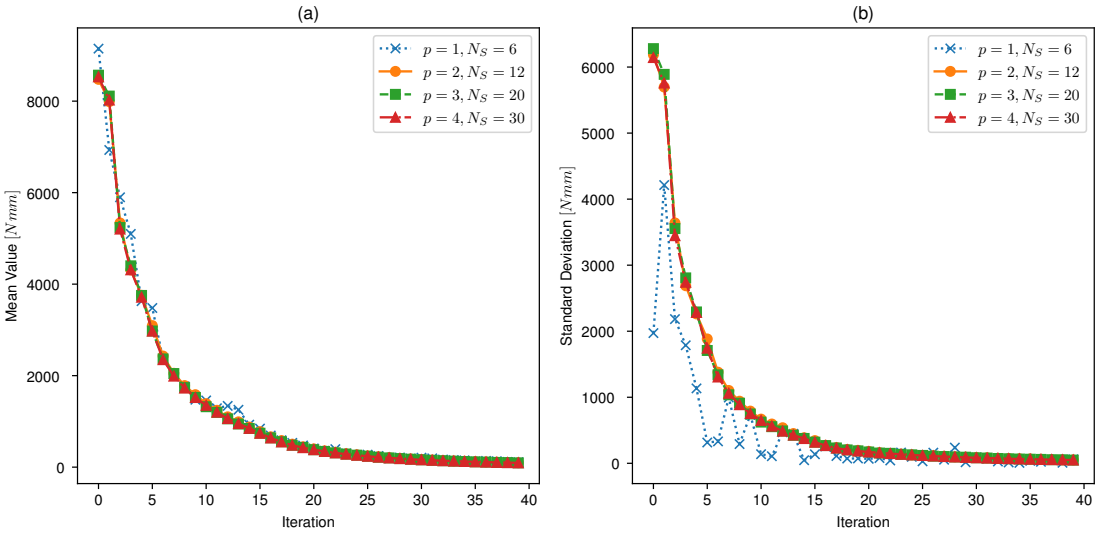


Figure 23 Convergence of the statistical moments of the mean optimization ($w_\mu = 1$ and $w_\sigma = 0$)

When optimizing only for the mean, Fig. 23 shows that for a polynomial degree of $p = 2$ and higher with corresponding samples, the convergence curves for both the mean and the variance are very similar. For $p = 1$ we can see, as already mentioned in Section 5.1.3, that there are slight differences in the mean (e.g., at iteration 12 and 13) and quite large deviations for the estimated standard deviation estimation. However, as the optimization progresses, even the inaccurate UQ estimations lead to the same geometry, and after iteration 30, the curves are almost equal. This is due to the fact that the mean values are estimated well enough. The final geometry is almost identical to the higher polynomial approximation, which will be shown in detail in the following sections.

p	N_s	$\mu[Nmm]$	$\sigma[Nmm]$	$\epsilon_\mu[\%]$	$\epsilon_\sigma[\%]$
1	6	108.9	6.04	10	11
2	12	97.2	51.8	4	5
3	20	93.5	48.5	0	1
4	30	93.4	49.2	-	-

Table 8 Comparison of final mean and standard deviation, as well as the relative difference compared to the most accurate estimation

Combined Mean and Standard Deviation Optimization

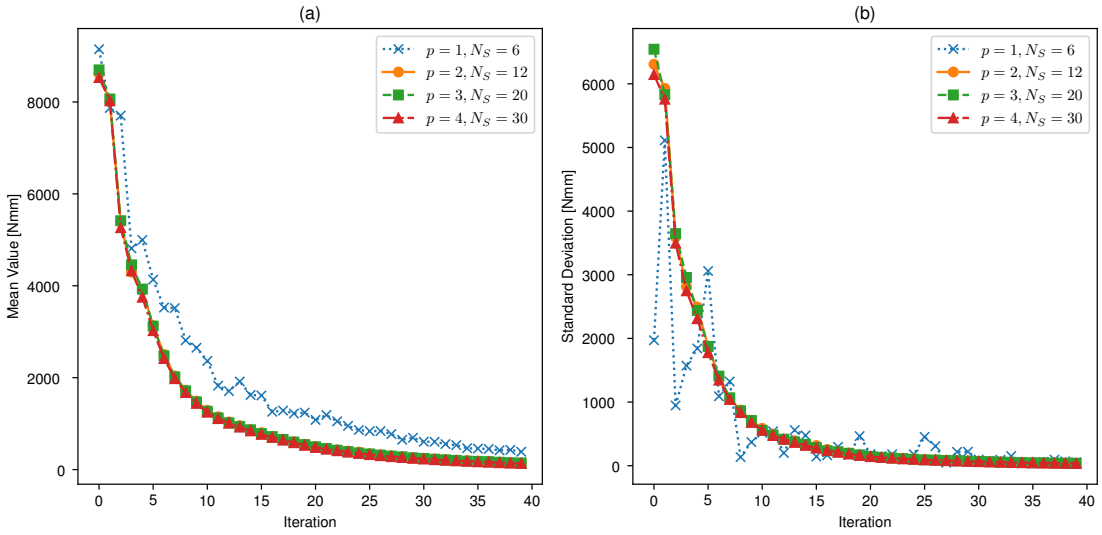


Figure 24 Convergence of the statistical moments of the combined mean and std robust optimization with weights $w_\mu = 1$ and $w_\sigma = 3$

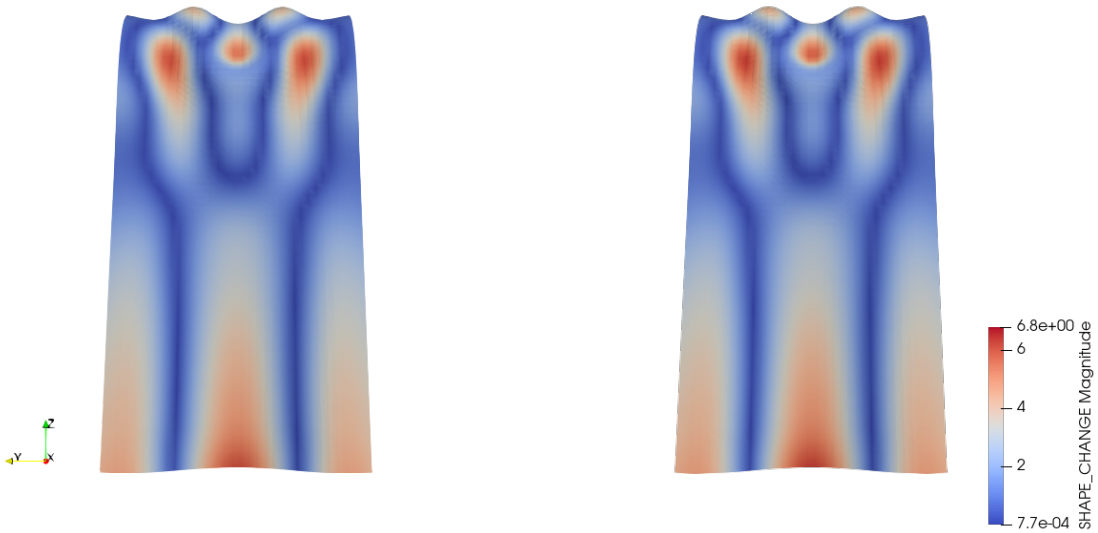
For the combined mean and standard deviation optimization, the poorly estimated standard deviation plays a much bigger role. Fig. 24 shows that the optimization for the mean is deviating quite a lot compared to the higher order approximations. The linear approximation leads to a worse design, as the final mean value differs by 192% and the standard deviation by 58%. We can also see that using $p = 4$ and $N_S = 30$ as initial values, we would have obtained an even better design by 9% in mean and 4% in standard deviation however, this results in a higher computational effort.

p	N_s	$\mu[Nmm]$	$\sigma[Nmm]$	$\epsilon_\mu[\%]$	$\epsilon_\sigma[\%]$
1	6	387.6	55.7	192	58
2	12	144.6	36.7	9	4
3	20	143.9	38.6	8	9
4	30	132.76	35.7	-	-

Table 9 Comparison of final mean and standard deviation, as well as the relative difference compared to the most accurate estimation

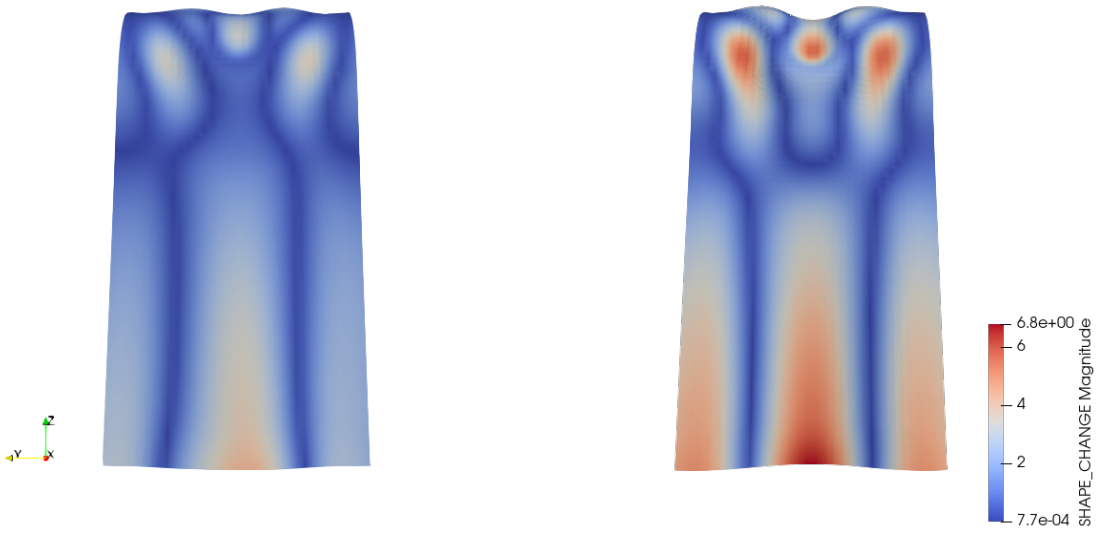
Shape

The difference of the approximation quality by the linear approximation between the mean only and the combined mean and standard deviation approximation can also be visualized when comparing the final geometries. When comparing Fig. 25a with 25b, we can see that – with very slight differences in shape change magnitude – the final geometries look the same. When setting Fig. 25c and 25d side by side, the differences are weighed bigger. The degree of indentation is much lower, and we can see at the bottom that the shape is folding outside for the linear approximation, whilst the higher order approximations (here $p = 2$) are folding to the inside.



(a) Mean optimized PCE Order = 1

(b) Mean optimized PCE Order = 2



(c) Mean and standard optimized PCE Order = 1

(d) Mean and standard optimized PCE Order = 2

Figure 25 Comparison of final geometries for different PCE orders.

5.2 Mass Minimization with Random Constraints

5.2.1 Problem Description

As a second application example, we present a structural optimization of a solid hook model. The hook has a height of $237mm$ and is modeled with a linear elastic material with a Young's modulus $E = 206.9GPa$ and a Poisson's ratio $\nu = 0.29$. The mass of to hook should be minimized while maintaining the initial compliance for two static load cases with a load at the center (LC1: $F = 32kN$) and tip ($LC2 : F = 16kN$) respectively. For both load cases, the hook is supported at the top. The initial weight of the hook is $241.8g$

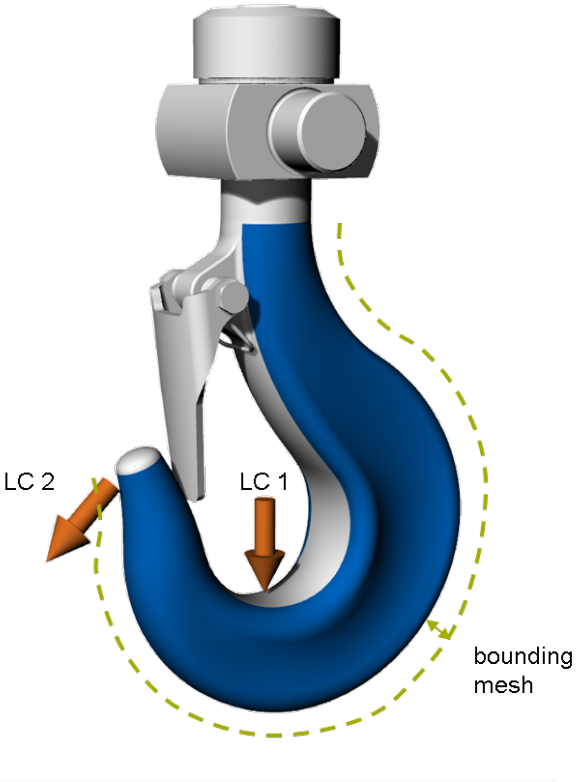


Figure 26 Optimization setup of the hook with the load-cases [39]

The mesh is discretized with 39292 nodes. However, only the nodes on the outer surface are considered for the shape optimization. These are 7117 nodes, resulting in 21351 design variables.

Optimization Formulation

The goal of the optimization problem is to reduce the mass, whilst retaining the stiffness of the structure with respect to the two different load cases.

Mathematically, the shape optimization problem can be formulated as:

$$\begin{aligned} &\min_{\mathbf{x}} \quad M(\mathbf{x}) \\ &\text{subject to} \qquad U_1(\mathbf{x}) = \frac{1}{2}\mathbf{u}_1^T\mathbf{K}(\mathbf{x})\mathbf{u}_1 \leq U_{1,0}, \\ &\qquad\qquad\quad U_2(\mathbf{x}) = \frac{1}{2}\mathbf{u}_2^T\mathbf{K}(\mathbf{x})\mathbf{u}_2 \leq U_{2,0}, \\ &\qquad\qquad\qquad\qquad \mathbf{x} \in \Omega_{\text{pack}}, \end{aligned} \tag{5.3}$$

The calculation of the mass response and its gradients takes about $0.5s$, and for each of the load cases approximately $4.9s$ with the *KratosMultiphysics StructuralAnalysisApplication* on an *Intel(R) Core(TM) i5-12450H* with 2.00 GHz. As the problem is now a constrained problem, the gradient projection algorithm is used. The filter function is linear with a radius of $25mm$. The optimization is terminated after 25 steps.

Modeling of Uncertainties

As in the first example, the uncertainties in this problem stem from the loading. For both load cases, the load direction is again varied. Again, the angles are modeled to follow a uniform distribution.

$$\begin{aligned} \xi_1 &\sim \mathcal{U}(-15, 15) \\ \xi_2 &\sim \mathcal{U}(-15, 15) \end{aligned} \tag{5.4}$$

The difference to the first example is that the objective function stays deterministic as the mass is independent of the load cases. The strain energy is now used as a constraint, making both constraints a random variable.

5.2.2 Deterministic Optimization

As the problem is now constrained, the gradient projection algorithm is used. The optimization is terminated after 25 steps.

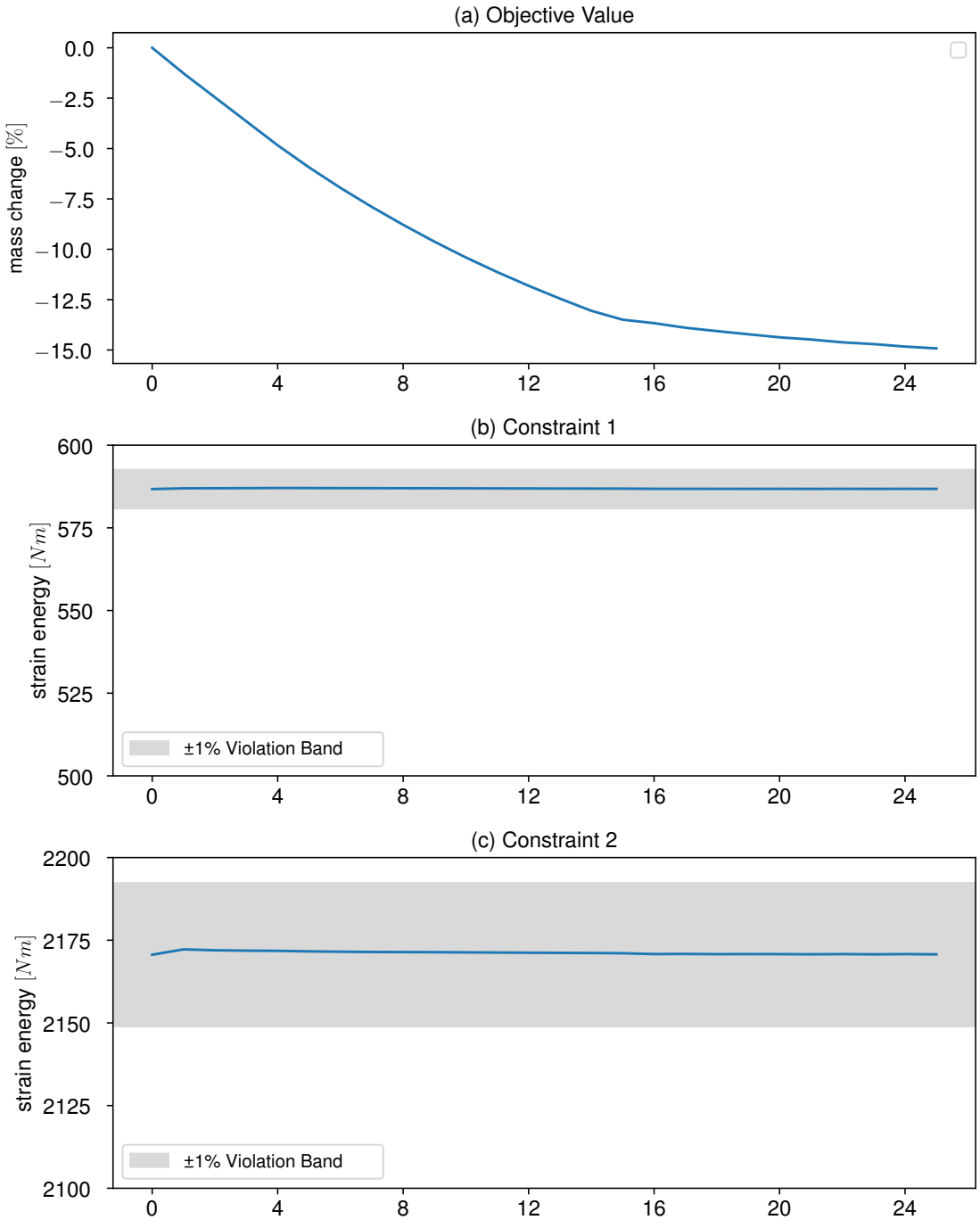
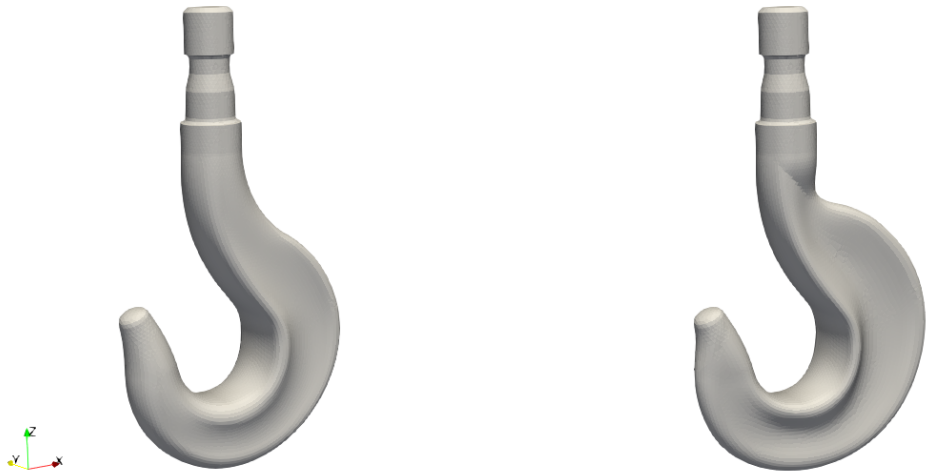


Figure 27 Convergence diagrams of the deterministic optimization

Fig. 27 shows the convergence behavior of the deterministic algorithm. After 25 iterations, the mass is reduced by 14.8%. We can see that both constraints stay within 1% violation during the whole process. Overall the convergence curve is smooth.

Fig. 28 shows the final geometry. We can see that the cross-section gets a bit wider and slimmer.



(a) Initial Geometry

(b) Geometry after 25 Optimization steps

Figure 28 Initial and final geometry after deterministic optimization

5.2.3 Uncertainty Quantification

As the second example is of a much higher dimension and two stochastic responses have to be considered, one cannot compute a reliable reference solution using many Monte Carlo samples. Because of this and limited computational time, we will use $p = 3$ and $N_S = 20$ for the PCE approximation. We will use the first iteration to obtain the reference statistics for the constraints,which are listed in Table 10.

Load case	**Deterministic Value** $[Nm]$	**Mean** $[Nm]$	**Standard Deviation** $[Nm]$
1	586	1534.5	617.15
2	2170.6	2338.4	273.7

Table 10 Statistical Moments for strain energy resulting from load cases 1 and 2

5.2.4 Robust Optimization

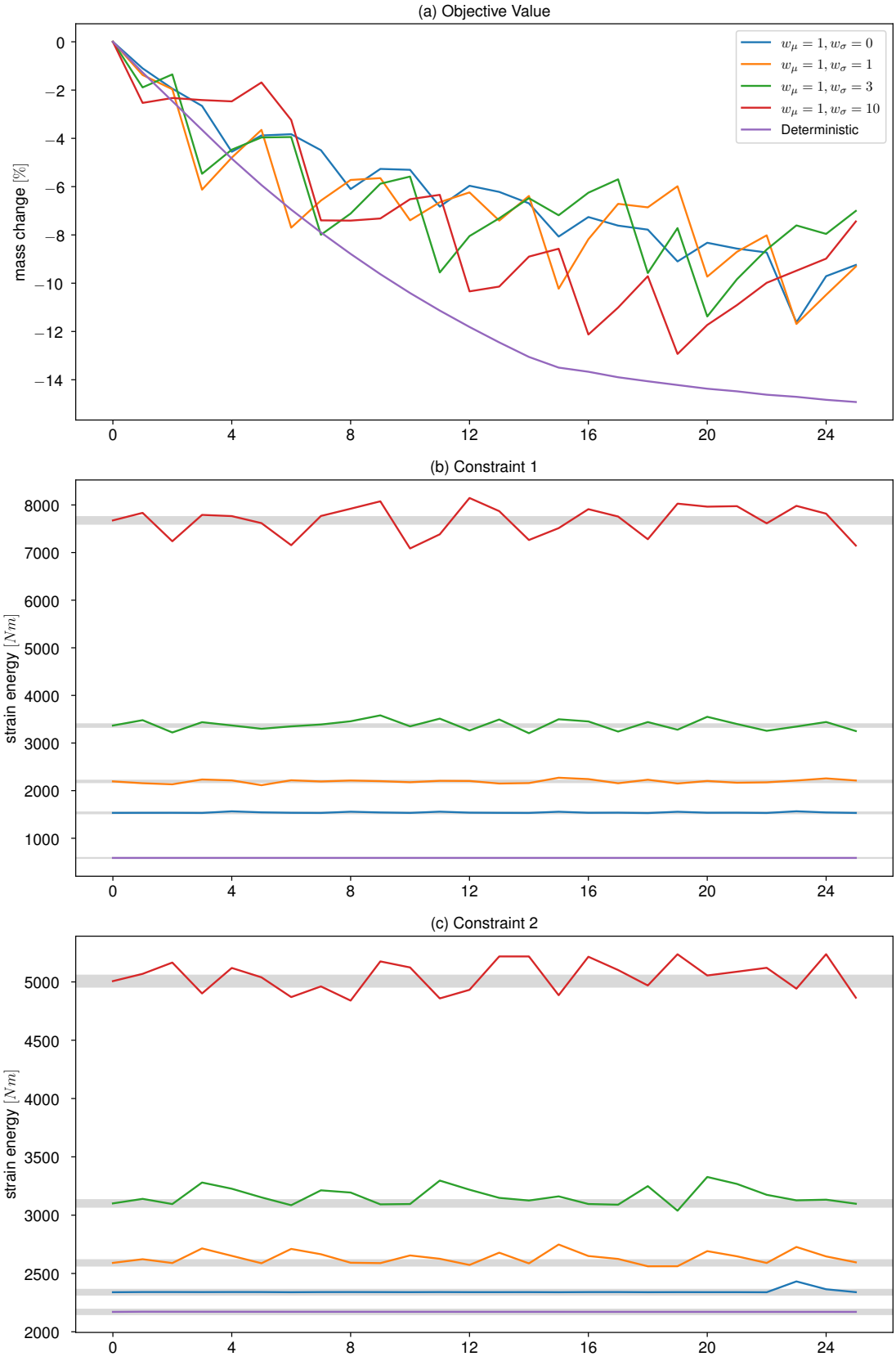


Figure 29 Convergence of the different robust optimizations. The grey bands visualize $\pm 1\%$ deviation from the initial constraint value

Fig. 29 shows the convergence curves of the robust optimizations with different weights. In the subfigure 29a, we can see that the objective function of the robust optimizations converges less smoothly than the deterministic one. This behavior is related to the constraint violations, which we can observe in subfigures 29b and c. Especially for the optimizations with a higher weight on the standard deviation, these violations are quite high. This so-called zig-zagging is quite common in gradient projection algorithms. The reason for this to occur is that the shape gradients are not accurate enough or that the step size is too big, both resulting in updates that violate the constraints, which need to be corrected in the next steps. The remedy for this would be to lower the step size (and thus use more steps) or to use more samples and a higher polynomial degree. Both result in a higher computational effort.

Geometry	**Weight** $[g]$	**Reduction** $[g]$	**Reduction** $[\%]$
Initial	241.8	0.0	0.0
Deterministic	205.7	36.1	14.93
$w_\mu = 1, w_\sigma = 0$	218.3	23.5	9.72
$w_\mu = 1, w_\sigma = 1$	219.3	22.5	9.31
$w_\mu = 1, w_\sigma = 3$	224.8	17.0	7.03
$w_\mu = 1, w_\sigma = 10$	223.8	18.0	7.45

Table 11 Weight of the hook after optimization

Table 11 shows the final weight for the different optimizations. The deterministic optimization is able to reduce the weight by $36.1g$. The weight for the robust design is minimized between $17.0g$ and $23.5g$.

Fig. 30 and 31 show the approximations of the distribution of the strain energy for both load cases after the optimization. As the responses are not minimized but just act as constraints on the problem, the differences are not as big as in the first example. We can see again a slight tendency that the deterministic optimisation has a wider spread than the robust results. Again, a trend of a lower mean with a higher spread to a higher mean with a lower spread can be seen with a stronger weighing of the standard deviation.

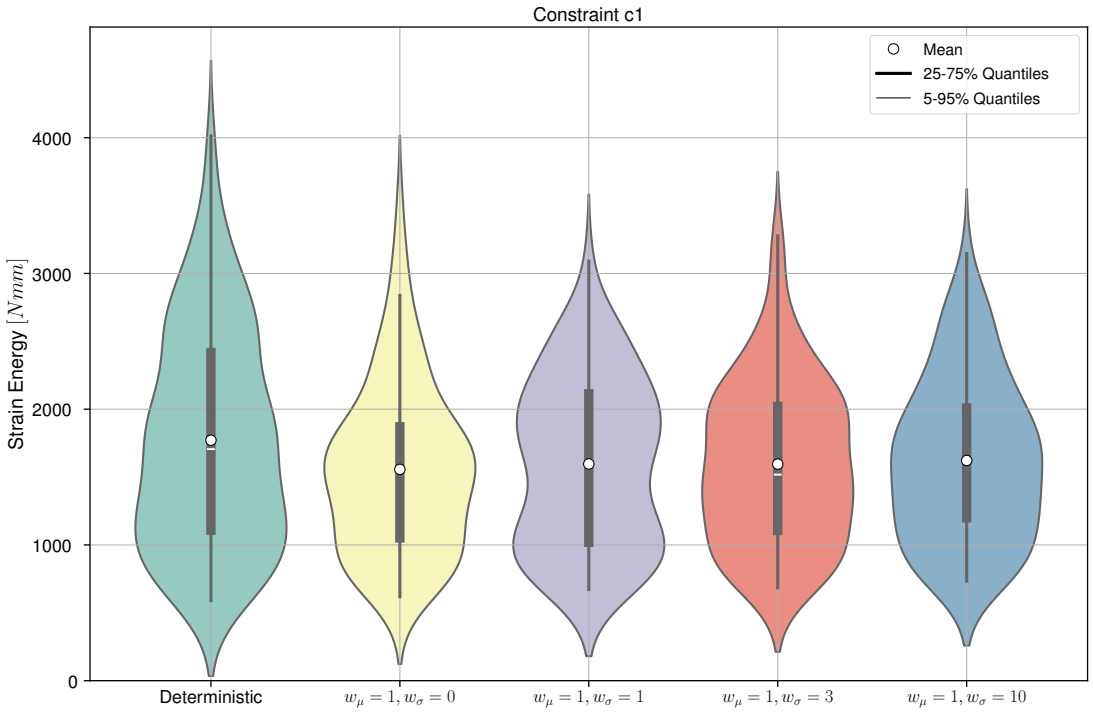


Figure 30 Violin plot of the strain energy resulting from load case 1

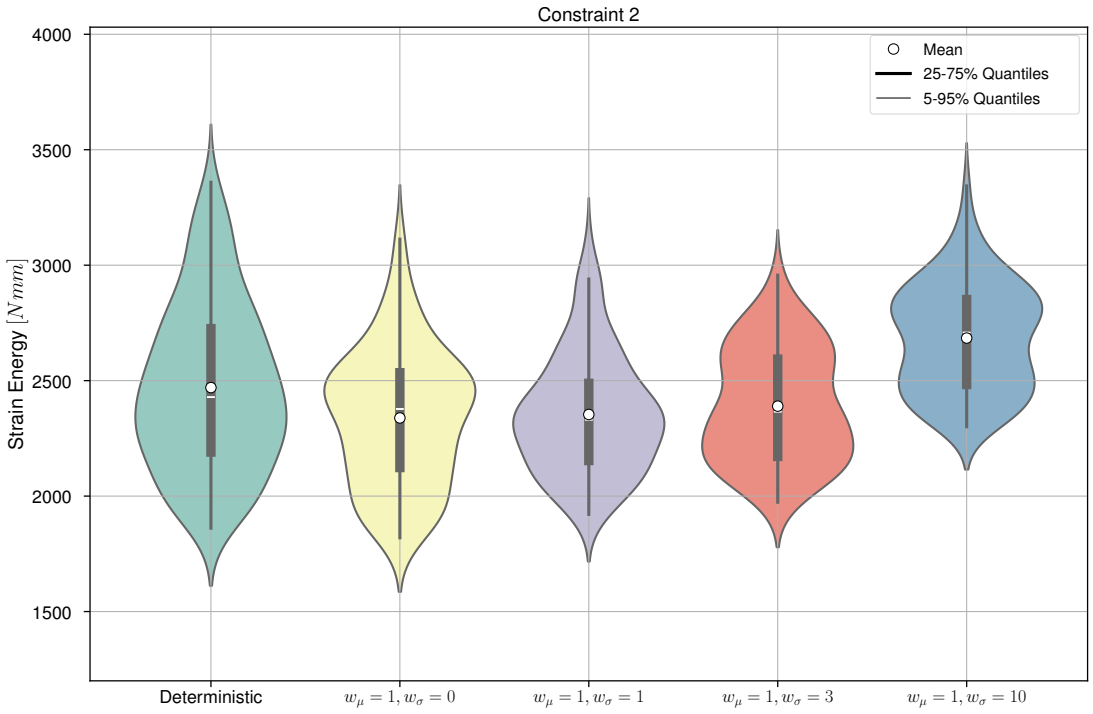


Figure 31 Violin plot of the strain energy resulting from load case 2

Tables 12 and 13 show the final statistical moments of the strain energy after the optimization. We can see that the statistical moments of the deterministic optimization are indeed higher than those in the initial configuration.

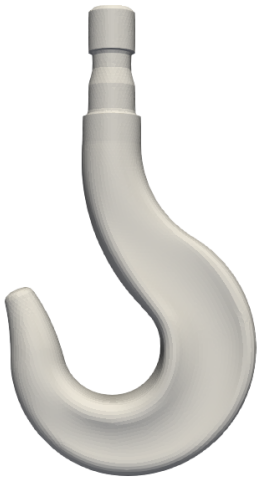
Optimization	**Mean** $[Nm]$	**Standard Deviation** $[Nm]$
Deterministic	1771.5	800.9
$w_\mu = 1, w_\sigma = 0$	1556.6	622.3
$w_\mu = 1, w_\sigma = 1$	1596.6	616.1
$w_\mu = 1, w_\sigma = 3$	1595.5	592.5
$w_\mu = 1, w_\sigma = 10$	1622.4	595.9

Table 12 Statistical Moments for strain energy resulting from load case 1

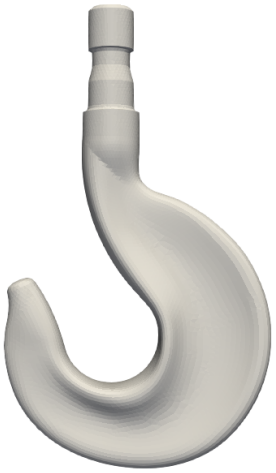
Optimization	**Mean** $[Nm]$	**Standard Deviation** $[Nm]$
Deterministic	2469.9	360.2
$w_\mu = 1, w_\sigma = 0$	2338.4	293.9
$w_\mu = 1, w_\sigma = 1$	2353.4	256.9
$w_\mu = 1, w_\sigma = 3$	2389.5	244.9
$w_\mu = 1, w_\sigma = 10$	2684.4	233.1

Table 13 Statistical Moments for strain energy resulting from load case 2

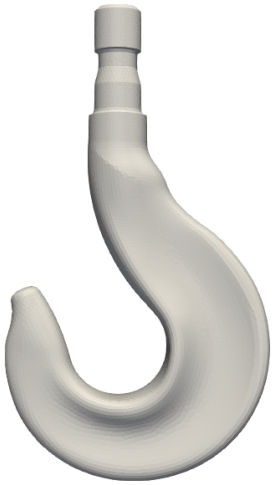
Shape



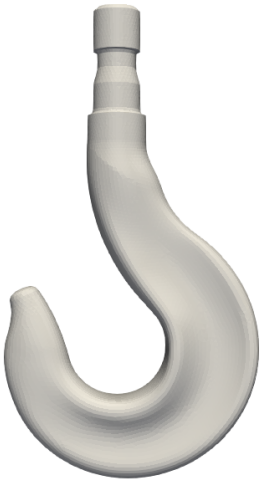
(a) Initial



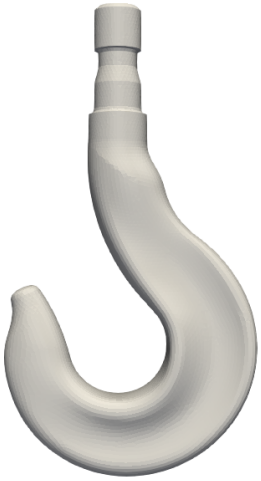
(b) Deterministic



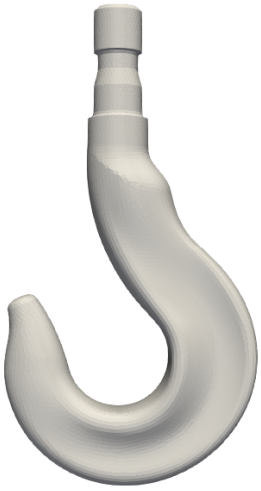
(c) $w_\mu = 1, w_\sigma = 0$



(d) $w_\mu = 1, w_\sigma = 1$



(e) $w_\mu = 1, w_\sigma = 3$



(f) $w_\mu = 1, w_\sigma = 10$

Figure 32 Initial and optimized hook shapes

When inspecting the final geometries, one can see that all shapes look quite similar. The main difference between the geometries is how far the right part is flattened. This difference can be made clearer in Fig. 33. It is also interesting to see that in the robust optimizations with higher weight on standard deviation more mass was shifted to the top of the hook (e.g., Fig. 32f). In contrast, for the deterministic optimization, mass and thickness were reduced at that part (see Fig. 32a).

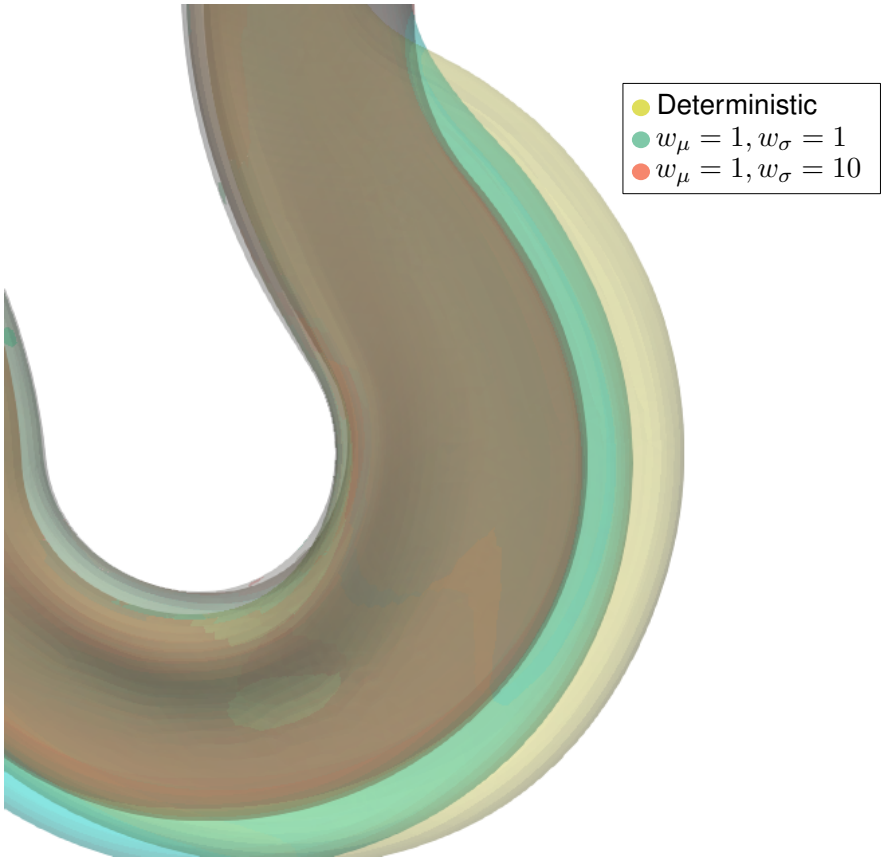


Figure 33 Detailed comparison between deterministic geometry and robust geometry

We can see that the robust optimization framework is able to produce designs that are different from the deterministic optimization process, both in performance and in shape. This does not necessarily mean that in the end the most robust design will almost be the most desired. However, it is an effective way to explore the design space further and to generate different designs to choose from as an engineer.

6 Conclusion

This thesis set out to answer the question of how a robust node-based shape optimization framework can be developed to account for uncertainties in design parameters while ensuring optimal and reliable performance. The research focused on integrating vertex morphing in combination with Monte Carlo simulations and Polynomial Chaos Expansion as uncertainty quantification techniques to handle the challenges of optimizing complex engineering structures under variable and uncertain conditions. By addressing the limitations of traditional deterministic optimization methods, which often overlook real-world uncertainties, this work aimed to provide a more resilient and dependable approach to shape optimization.

The results of the analysis demonstrated that the proposed framework effectively addresses the research question by integrating uncertainty quantification into the optimization process. The case studies, including the randomly loaded arch structure and the mass minimization of a hook under variable load conditions, showed that the robust optimization framework can lead to designs that outperform the results of deterministic methods. Designs obtained through the robust optimization process displayed improved performance in the presence of uncertainty. Moreover, the application of PCE allowed for an efficient and accurate estimation of statistical moments, enabling the optimization framework to account for both mean performance and variability. The second example showed, however, that the robust framework using PCE might lead to convergence problems if the accuracy of the estimated gradient is not high enough.

Furthermore, some limitations and deficits in the analysis should be acknowledged. Firstly, while the framework successfully handled the chosen academic case studies, the complexity of real-world problems might introduce additional challenges that were not fully explored in this work. The computational cost of uncertainty quantification, particularly in high-dimensional problems, remains a critical concern. Although PCE offers faster convergence compared to Monte Carlo methods, the accuracy of results still depends on the number of polynomial terms and samples used, and the selection of these parameters can be problem-dependent. Finally, the case studies relied on simplified assumptions for uncertainty modeling, such as uniform distributions for load variability, which might not always reflect the full range of uncertainties in real-world applications.

The outlook for further research and development in this field is promising. One area that warrants exploration is the application of this robust optimization framework to more complex, real-world engineering problems, such as aerospace and automotive structures, where uncertainty in load conditions, material properties, and environmental factors is common. Additionally, future work could focus on enhancing the computational efficiency of uncertainty quantification methods. While PCE showed potential in reducing computational cost, hybrid approaches that combine PCE with other uncertainty quantification techniques or machine learning-based surrogate models may offer a way to further accelerate the process, especially for high-dimensional problems.

Coupling shape optimization with other advanced manufacturing techniques, such as topology optimization or additive manufacturing constraints, could lead to new design paradigms that are not only robust but also optimized for manufacturability. Moreover, incorporating more realistic uncertainty models, such as stochastic material properties or time-dependent variability, could enhance the practical relevance of the framework in real-world applications.

Finally, integrating robust optimization techniques with multi-disciplinary optimization (MDO) frameworks, where multiple physics, such as fluid-structure interactions or thermal stresses, are considered, could further broaden the scope and impact of this research. Such advancements would make robust shape optimization an indispensable tool for industries that demand high-performance and reliable designs under uncertainty.

List of Figures

List of Tables

Bibliography

[1] Majid Hojjat. *Node-based parametrization for shape optimal design*. PhD thesis, Technische Universität München, 2015.

[2] Majid Hojjat, Electra Stavropoulou, and Kai-Uwe Bletzinger. The vertex morphing method for node-based shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 268:494–513, 2014.

[3] Christian Sabater Campomanes. Development of efficient surrogate-assisted methods to support robust design of transonic wings. Technical report, DLR e.V., 2022.

[4] Dinesh Kumar, Mehrdad Raisee, and Chris Lacor. *Combination of Polynomial Chaos with Adjoint Formulations for Optimization Under Uncertainties*, pages 567–582. Springer International Publishing, Cham, 2019.

[5] Wei ZHANG, Qiang WANG, Fanzhi ZENG, and Chao YAN. A novel robust aerodynamic optimization technique coupled with adjoint solvers and polynomial chaos expansion. *Chinese Journal of Aeronautics*, 35(10):35–55, 2022.

[6] Kai-Uwe Bletzinger. Lecture notes in structural optimization, February 2023.

[7] Jamshid A Samareh. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA journal*, 39(5):877–884, 2001.

[8] J. A. Nelder, John A. Nelder, R. Mead, Roger Mead, R. Mead, R. Mead, and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 1965.

[9] Kit-Sang Tang, Kim-Fung Man, Sam Kwong, and Qianhua He. Genetic algorithms and their applications. *IEEE signal processing magazine*, 13(6):22–37, 1996.

[10] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1:3–52, 2002.

[11] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[12] A. Galántai. The theory of newton's method. *Journal of Computational and Applied Mathematics*, 124(1):25–44, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[13] C. G. BROYDEN. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 03 1970.

[14] Jo Bo Rosen. The gradient projection method for nonlinear programming. part i. linear constraints. *Journal of the society for industrial and applied mathematics*, 8(1):181–217, 1960.

[15] Kai-Uwe Bletzinger. A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape. *Structural and Multidisciplinary Optimization*, 49:873–895, 2014.

[16] Michael Havbro Faber. *Statistics and probability theory : in pursuit of engineering decision support*. Springer Science+Business Media, New York, 2012.

[17] S.M. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Elsevier Science, fifth edition, 2014.

[18] Reuven Y. Rubinstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method*. Wiley Publishing, 3rd edition, 2016.

[19] Michael Schick. *Uncertainty Quantification for Stochastic Dynamical Systems: Spectral Methods using Generalized Polynomial Chaos*. PhD thesis, 2012.

[20] Russel E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49, 1998.

[21] Remi Leluc, Francois Portier, and Johan Segers. Control variate selection for monte carlo integration. *Statistics and Computing*, 31(4), jul 2021.

[22] Norbert Wiener. The Homogeneous Chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

[23] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.

[24] Chris Lacor and Eric Savin. *General Introduction to Polynomial Chaos and Collocation Methods*. 01 2018.

[25] Jeroen A. S. Witteveen and Hester Bijl. Efficient quantification of the effect of uncertainties in advection-diffusion problems using polynomial chaos. *Numerical Heat Transfer, Part B: Fundamentals*, 53(5):437–465, 2008.

[26] Arun Kaintura, Tom Dhaene, and Domenico Spina. Review of polynomial chaos-based methods for uncertainty quantification in modern integrated circuits. *Electronics*, 7(3), 2018.

[27] Bert Debusschere. *Intrusive Polynomial Chaos Methods for Forward Uncertainty Propagation*, pages 617–636. Springer International Publishing, Cham, 2017.

[28] Robin Schmidt, Matthias Voigt, and Ronald Mailach. *Latin Hypercube Sampling-Based Monte Carlo Simulation: Extension of the Sample Size and Correlation Control*, pages 279–289. Springer International Publishing, Cham, 2019.

[29] J.M. Hammersley and D.C. Handscomb. *Monte Carlo Methods*. Methuen's monographs on applied probability and statistics. Methuen, 1964.

[30] I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[31] Christine Louise Schmid. *Generalization of polynomial chaos for estimation of angular random variables*. PhD thesis, Missouri University of Science and Technology, 2020.

[32] Sebastian Burhenne, Dirk Jacob, and Gregor Henze. Sampling based on sobol' sequences for monte carlo techniques applied to building simulations. pages 1816–1823, 01 2011.

[33] A. del Junco and J. Michael Steele. Hammersley's Law for the Van Der Corput Sequence: An Instance of Probability Theory for Pseudorandom Numbers. *The Annals of Probability*, 7(2):267 – 275, 1979.

[34] Serhat Hosder, Robert Walters, and Michael Balch. Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables. 125, 04 2007.

[35] Serhat Hosder, Robert Walters, and Rafael Perez. *A Non-Intrusive Polynomial Chaos Method For Uncertainty Propagation in CFD Simulations*.

[36] Fabio S Dias and Gareth W Peters. Option pricing with polynomial chaos expansion stochastic bridge interpolators and signed path dependence. *Applied Mathematics and Computation*, 411:126484, 2021.

[37] Mattias Enstedt and Niklas Wellander. Uncertainty quantification of radio propagation using polynomial chaos. *Progress In Electromagnetics Research M*, 50:205–213, 2016.

[38] Vicente Mataix Ferrándiz, Philipp Bucher, Rubén Zorrilla, Suneth Warnakulasuriya, Riccardo Rossi, Alejandro Cornejo, jcotela, Carlos Roig, Josep Maria, tteschemacher, Miguel Masó, Guillermo Casas, Marc Núñez, Pooyan Dadvand, Salva Latorre, Ignasi de Pouplana, Joaquín Irazábal González, AFranci, Ferran Arrufat, riccardotosi, Aditya Ghantasala, Klaus Bernd Sautter, Peter Wilson, dbaumgaertner, Bodhinanda Chandra, Armin Geiser, Inigo Lopez, lluís, jgonzalezusua, and Javi Gárate. Kratosmultiphysics/kratos: Release v9.5, April 2024.

[39] Armin Geiser, Ihar Antonau, and Kai-Uwe Bletzinger. Aggregated formulation of geometric constraints for node-based shape optimization with vertex morphing. pages 80–94, 01 2021.

The software package, KratosMultiphysics, used in this work is open-source and available for download at *https://github.com/KratosMultiphysics/Kratos.* The source code for replicating the results can be found on *https://github.com/PhilippLeoJakobs/Kratos/tree/Robust_SO.* Please contact the author for the datasets used to generate the results.

Declaration

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. In addition, I declare that I make the present work available to the Chair of Structural Analysis for academic purposes and in this connection also approve of dissemination for academic purposes.

Munich, 14.10.2024, Signature