Department of Construction Sciences
Solid Mechanics

ISRN LUTFD2/TFHF-26/5277-SE(1-74)

Bayesian Shape Optimisation

A Surrogate Modelling Approach to Shape Optimisation

Master's Dissertation by

Anton Bäckström & Adam Thell

Supervisors:
Prof. Dr. Mathias Wallin, LTH
Dr. Gunnar Granlund, LTH
Erik Niska, Tetra Pak
Karolina Wamsler, Tetra Pak

Examiner:
Dr. Håkan Hallberg, LTH

Printed by Media-Tryck AB, Lund, Sweden
For information, address:
Division of Solid Mechanics, Lund University, Box 118, SE-221 00 Lund, Sweden
Webpage: www.solid.lth.se

Abstract

As environmental demands increase, the requirements for the design and production of aseptic packaging increase as well. To meet these new requirements, an optimisation framework is to be developed, but traditional - gradient based - approaches struggle to handle the non-smooth nature of the problem. Instead, this thesis aims at building a framework where the underlying problem is considered a black-box function. Convex optimisation with polynomial regression was initially evaluated as an alternative to traditional approaches, but ultimately the convex assumption proved inadequate. Bayesian optimisation with Kriging modelling was then explored as an alternative able to handle these issues. Three different initial sampling methods - Latin Hypercube sampling, Hammersley sequence sampling, and Sobol sequence sampling - were tested. Accompanying these, three different acquisition functions - Expected Improvement, Upper Confidence Bound, and Parallel Expected Improvement - were tested.

The initial testing showed that the commercial software HyperWorks' implementation of Kriging modelling was inadequate; the modelling itself performed poorly, and the framework was not adaptable. Instead, a self-written Baysesian optimisation loop with Kriging modelling, based on BoTorch, was implemented. The methodology was able to perform shape optimisation and improve designs given a desired response in form of a pressure distribution. Hammersley sequence sampling proved to be a consistent method that performed well. Latin hypercube sampling proved not as consistent as Hammersley, but was able to outperform Hammersley in some runs. Sobol sequence sampling performed poorly in the testing, but from the conditions no conclusions could be drawn from this. Expected Improvement proved unreliable in early testing, and was thus discarded as an acquisition function. Parallel Expected Improvement and Upper Confidence Bound were both able to improve the model.

Keywords: Bayesian Optimisation, Surrogate modelling, Kriging modelling, Polynomial regression, Sensitivity analysis, Design of experiments, Sampling, Sobol, Hammersley, Latin Hypercube, Structural Optimisation, Shape Optimisation, B-splines, Ultrasonic Sealing

Using Surrogate Models for Shape Optimisation

How can we optimise the design of a machine component when the problem is too complex for traditional methods? In this thesis we explore a surrogate based approach for solving these problems.

Traditionally when designing a new component, an initial design is made and the component's performance is measured. Most likely, the first version does not perform as required, and the design is changed based on the initial tests. When the constructor has designed the second version of the component, its performance is once again tested, and once again redesigned if needed. This loop continues until the performance of the part is satisfactory. This process can be time consuming and can take many iterations before the designed part performs satisfactory. Our thesis will attempt a different approach of designing a component. Instead of designing a component and seeing what its performance is, we do the opposite; we determine a satisfactory performance and use optimisation to find the design that yields that performance. The framework we will implement is based on Bayesian optimisation. Bayesian optimisation consists of three main parts: an initial sampling, a surrogate model, and an acquisition phase.

The first part of Bayesian optimisation is sampling methods. Sampling evenly in one dimension is simple, but in multiple dimensions it is not as simple. Spacing points evenly, while not reusing previous coordinates, becomes more difficult as the amount of dimensions increase. Three different sampling methods, all aiming at spacing points evenly while not reusing coordinates, have been explored and tested throughout this work. Relevant literature state the different sampling methods all have advantages and disadvantages. By varying parameters and testing the performance of the sampling methods, we were able to to verify some of the advantages and disadvantages from literature, while finding others relevant to the case at hand. In the end, two of the methods performed well, while the third underperformed based on the expectations set by literature.

The second, and most important, part of Bayesian Optimisation is surrogate models. A surrogate model is a model, or functional form, that a unknown function is assumed to take. This can be a e.g a polynomial or a Gaussian process. A polynomial is too simple to work in many applications. Instead, a Gaussian process is a good choice since Bayesian Optimisation is built on statistics. Our testing showed a Gaussian process is able to adequately interpolate a function, verifying it is a good choice for a surrogate model.

The last part of Bayesian Optimisation is acquisition functions. They provide a good way improving the surrogate model while lessening the computational cost, when the unknown function is expensive to evaluate. Three different acquisition functions were tested in this thesis. By only valuing high values these functions improve the surrogate model in regions of interest, disregarding regions of low values. One of the functions, despite being the most popular in the scientific community, proved too inflexible and easily got stuck, signifantly increasing the computational cost of the algorithm. The other two had different work arounds, but both yielded similar results, significantly improving the model.

The thesis shows that it is possible to use a surrogate based approach to solve problems too complex for traditional structural optimisation methods. Although, the methodology created is not complete and needs to be tested more thouroughly and tuned to consistently yield good results.

Acknowledgements

We would like to express our gratitude to our main supervisor Mathias Wallin for providing support and guidance, and helping us maintain focus on what is important throughout the project. Much gratitude is directed towards our co-supervisor Gunnar Granlund for providing us with the opportunity to work on our thesis with Tetra Pak in the first place. Further, he has helped us by providing invaluable guidance at the start of our work and last but not least, being an excellent teaching assistant throughout our master's programme. Furthermore, we would like to express our gratitude to industry co-supervisor Erik Niska for always having an abundance of ideas, entrusting us with free reins to develop and discover the right path forward, and helping us think like an industry engineer. Karolina Wamsler stepped up as industry co-supervisor and we are enormously grateful for her help, providing much needed help with the quirks (and features) of the software. Last but not least, Johan Lindström at mathematical statistics, despite not being our supervisor, has perhaps influenced this work the most. As we realised this thesis was a work of statistics applied on solid mechanics, and not a thesis on solid mechanics with elements of statistics, Johan gave us invaluable insight, helped us understand what we needed to research through interesting meetings, and was a great e-mail correspondent whenever questions arose.

Disclaimers

Wikipedia has been cited as a source throughout this work. At no point has it been used as a primary source, nor has any information only been acquired there. However, the formulations there often present the information more accessibly than the primary sources, making it a good complementary source for the interested reader, as it has been for us.

A.I has been used throughout this thesis as a research assistant, finding us sources to study, as well as helping us sort out certain coding bugs. A.I has not been used to analyse results, draw conclusions, or write any part of the report. Although, at times, awkward sentences have been identified with the feedback of A.I.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This master's thesis project has been conducted at the division of Solid Mechanics at LTH, Lund university in collaboration with the Sealing Applications and Modelling group at Tetra Pak in Lund. Tetra Pak is a global leader in the packaging industry, designing and producing both packaging as well as filling machines. The company was founded in 1951 in Lund and as of 2026, is selling to over 160 countries. Having introduced the first package, later named Tetra Classic, to the market in 1952, the company spent the 1950's improving the technology, resulting in the release of an aseptic version in 1961. The aseptic technology allows for food to be stored at room temperature for up to a year, despite normally expiring after weeks in chilled storage. Being able to greatly expand the shelf life of food products, and lowering storage and distribution costs, the aseptic technology has been called most important innovation in food packaging of the 20th century by the Institute of Food Technologists.

Continuing to innovate, Tetra Pak now aims at further improving the packaging. Integrating optimisation methods in the design process can shorten the time to market and improve the final results. Traditional methods for structural optimisation are gradient-based, which work well for problems with continuous gradients. For non-continuous gradients, these methods are difficult to implement. The aim of this thesis is therefore to attempt the implementation of another framework, which disregards the non-continuous gradients, called Bayesian Optimisation, and perform a proof of concept using shape optimisation.

1.1 Environmental Challenges in Packaging

As demands on sustainability and environmental concerns increase, manufacturing industries face new challenges. To continue the work for increased sustainability, and in light of a recent EU regulation, Tetra Pak will need to reconsider the design of their aseptic line of packaging. The regulation in question is EU Regulation 2025/40 of the European Parliament and of the Council of 19 December 2024 on packaging and packaging waste. It tasks member states with recycling a minimum amount of percentage by weight of listed materials in packaging. In Table 1 of Annex II to Article 6, cardboard with aluminium liners is listed under category 3, "Composite packaging of which the majority is paper/cardboard". According to paragraph 3 of Article 6, all packing as described in Table 1 must be recyclable in accordance with grades A, B and C as specified in Table 3 of Annex II. Table 3 states that packaging of grade C, where less than 70% by weight is recyclable, will not be allowed on the market by 2030. Further, grades below B, where less than 80% of the packaging by weight is recyclable, is not allowed to be put on the market by 2038 [7]. An overview of these numbers can be seen on a timeline in figure 1.1. Simplified, the regulation signals that packaging must be made simpler, both in terms of constitution and recycling.

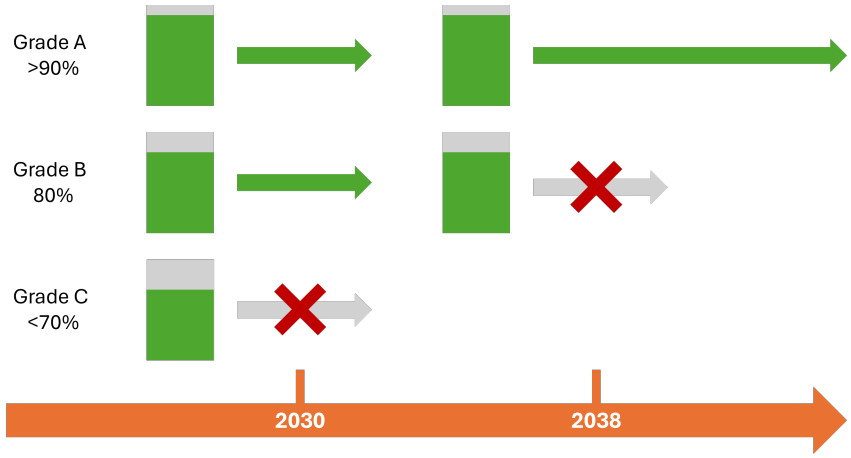


Figure 1.1: Timeline of the EU regulation regarding increased standards of percentage of recyclable materials in packaging.

Due to challenges in recycling and separation of aluminium from the polymer in cardboard packaging, Tetra Pak aims at removing aluminium foil from their aseptic line of packaging. Removing the aluminium liner from the packaging works towards the goal of eliminating aluminium waste. However, it comes with difficulties in the packaging production process. Currently, the sealing of the packaging utilises induction heating to melt the polymer coating of the packaging material, which - after cooling - seals the package. Without aluminium foil, this method is no longer viable. Possible replacements for the induction based method are direct heating or ultrasonic based methods. Ultrasonic sealing methods are currently only in use for chilled products, where quality standards can be lower. The change to ultrasonic sealing for aseptic products requires new machine components with higher sealing performance to meet the higher quality standards associated with these packages, as well as fit in the smaller footprint available in the filling machine. To increase the performance of these machine parts and decrease their footprint, Tetra Pak aims to explore to possibility of incorporating an efficient optimisation methodology into the design process.

1.2 Packaging and Sealing at Tetra Pak

In order to familiarise the reader with the problem the methodology developed is applied to, a description of the packaging process and the ultrasonic transversal sealing process at Tetra Pak are described below.

1.2.1 Packaging Process

In an aseptic filling machine, the process of turning a roll of packaging material into finished packages has a number of general steps. Accompanying this section, figure 1.2, visualises these steps that are related to the sealing processes. Firstly, a roll of flat packaging material is unwound at the far end of the filling machine. To the unwound paper material, a polymer strip is applied in the direction of travel. After the packaging material has been sterilised in the machine, this strip seals the inside of the packaging from the outer seal as the material is formed into a tube and sealed longitudinally. The longitudinal seal is created using either induction heating or hot air sealing methods. During the sealing process the tube is continuously filled with product.

After filling, the tube needs to be sealed and split into individual packages. This is achieved with a transversal seal, which utilises a double action, sealing both the top of the current package as well as the bottom of the subsequent package at the same time. This seal is made using either induction heating or ultrasonic sealing. Since the time allocated for this step is a fraction of a

second, direct heating methods are too slow. It is the ultrasonic method for the transversal seal that the methodology developed in this thesis will be applied to.

After the transversal seal has been created, and while the sealed areas are still held fast, a knife splits the two transversal seals created, separating the package from the tube. Then, folding steps are conducted, resulting in a completed, product-filled, package. In the following section, the ultrasonic sealing method, as well as its mechanisms and requirements are detailed further.

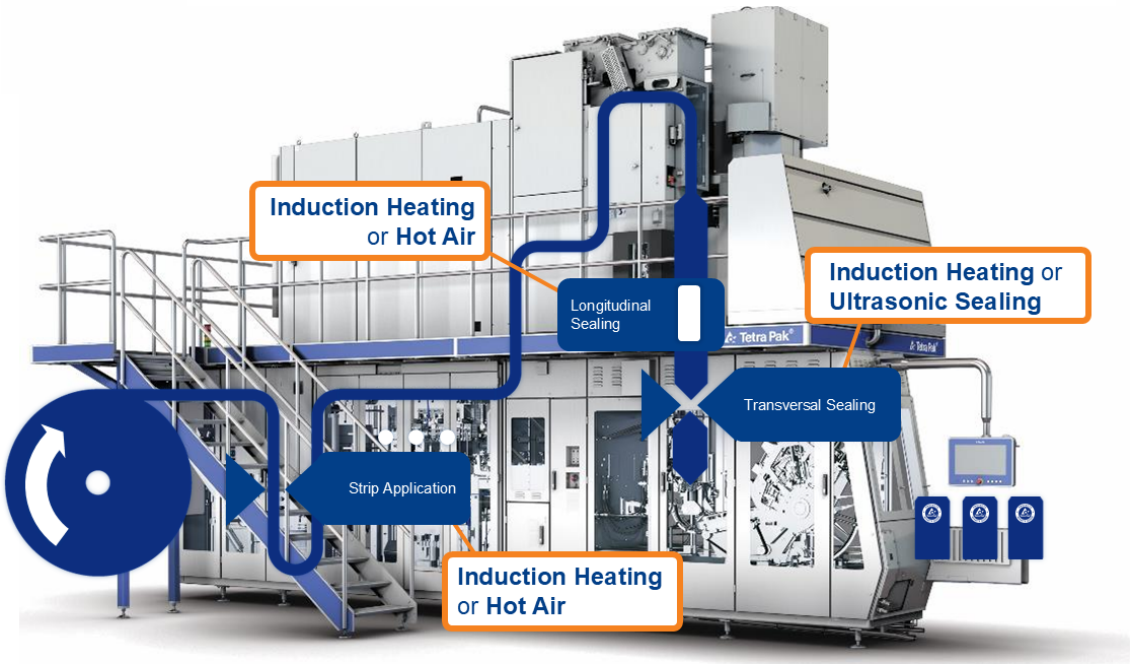


Figure 1.2: Flow of packaging material from roll to finished package through filling machine with sealing related steps marked.

1.2.2 Ultrasonic Sealing

The principle of ultrasonic sealing in the context of Tetra Pak consists of an assembly with a frequency generator converting 50 Hz to the desired frequencies, in the region above 20 kHz. This is fed into a piezo-electric unit, transforming the electric energy to mechanical energy, which causes one of the two sealing elements, the sonotrode, to vibrate. The other sealing element is called an anvil, and is the focus of this thesis. The two components are marked in figure 1.3, showing how they are in contact with the two layers of packaging material.

In order to create a seal, two conditions need to be fulfilled. Firstly, sufficient heat needs to be generated in order to melt the polymer sheet of the packaging material, which acts as the "glue" of the seal. Secondly, sufficient pressure needs to be applied during the whole process, partly to ensure that the packaging material does not move or shift during either heating or cooling, and partly to ensure that the melted polymer is distributed correctly in the sealing region.

The steps of the ultrasonic sealing method for the transversal seal are explained here. First, the whole assembly of sealing elements and packaging material is compressed, or pre-loaded, fulfilling one of the two conditions for sealing. The preloading compresses the packaging material and voids product from the sealing region, marked ”Contact zone” in figure 1.3. Then, the piezo-electric unit vibrates the sonotrode. These vibrations are transferred to the packaging material, which in turn is rapidly compressed and decompressed. The action of compression and decompression of

the packaging material causes the fibres in the paperboard to rub against each other, resulting in friction, generating heat. Thus, fulfilling the other condition of the sealing process. The heat generated in the material causes the polymer sheet of the packaging material to melt. After a certain pulse of vibrations, enough heat has been applied and the polymer is sufficiently melted. The vibrations cease and the polymer is allowed to cool and recrystallise, still under pressure from the preloading, creating a seal. After separating the packages with the knife, the pre-loading is released.

Since the pressure from the preloading is one of the key factors in the sealing process and dictates where heat is generated, the pressure distribution in the sealing region is vital. Factors such as a varying number of layers of packaging material along the seal due to an overlap from the longitudinal seal, as well as evacuating product from the sealing region must be taken into account. The anvil is one of the key components affecting the pressure distribution, and must be shaped in the correct way. Currently, this is a challenge which includes clever engineering guesswork through trial and error in combination with simulations. Tetra Pak currently has an idea for the desired pressure distribution, but the subject is open to research, and a method that is adaptable to changes in desires and simulation complexity is wanted.

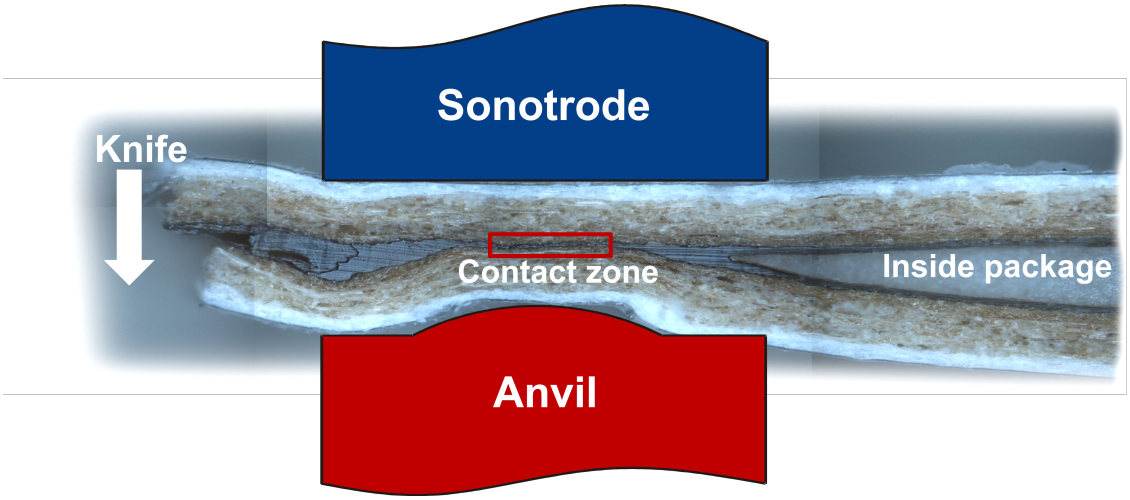


Figure 1.3: Cross section of a transversal seal, with two layers of packaging material compressed by a sonotrode and anvil. The location of the cut is marked with "knife". Product is marked with "Inside package".

1.3 Scope and Aim

The aim of this project is to develop an optimisation methodology using Bayesian optimisation and start implementing this methodology on the design of an ultrasonic sealing anvil, see section 1.2.1. Additionally, this report will act as a literature study for further development of the methodology at Tetra Pak for potential future thesis workers.

To achieve this aim, the primary scope was limited to outlining how an efficient optimisation methodology can be constructed, and develop an optimisation method using the tools available at Tetra Pak. The outline will detail the different steps of Bayesian optimisation, and explain the theory behind them. An important part of the methodology is ensuring the different tools available can communicate, it is therefore necessary to ensure that these different software programs can be integrated into a single chain. This optimisation method will then be applied to the ultrasonic sealing anvil in a simplified 2D design case. The aim of this 2D case is to obtain a sealing anvil yielding an arbitrary contact pressure curve between the inner surfaces of the two

pieces of paperboard, see figure 1.3. Academically, the scope of the project is to, in part, increase our own knowledge as engineers in subjects not covered in courses, as well as apply theory and help optimise technology with ultrasonic sealing for Tetra Pak.

1.4 Previous Work

Projects featuring optimisation have earlier been performed in collaboration with Tetra Pak. Håkansson and Lychou conducted one such project concerning size optimisation of the Bridge Twist membrane found in Tetra Top packaging [11]. Part of the aim of their thesis was to lower the cost and reduce the time needed for the design process of a specific component. Håkansson and Lychou employed an optimisation workflow similar to the one we performed in our initial testing. Throughout, they used Altair's software package, a package we found not to be optimal for our case, and used a DOE to construct their surrogate model. Their sampling was performed using i.a full-factorial and Hammersley sampling. They then optimised on the surrogate model using sequential quadratic programming. The conclusion of the thesis was that their model differed from the input values.

Although they used similar methods to us, the aim of their thesis differs from ours, and they put focus on refining material models. The results and conclusion of their thesis is therefore of little relevance to our work.

Chapter 2

Structural Optimisation

2.1 Principles of Structural Optimisation

The fundamental idea of structural optimisation is to produce a better design given certain conditions. It starts with defining the component design as a function of design variables as well as an objective function, representing the sought after properties of the design. The design variables are then updated in such a way as to find a better value of the objective function, implying a better design of the component, though only in regard to the sought after property. In structural optimisation, such objective functions and properties can be e.g. minimising compliance, minimising the maximum stress in the component, or minimising the volume or mass of the component. Additionally, constraints can also be added to the optimisation. These represent certain conditions that the design must fulfil. Examples can be setting an upper limit on weight or effective stress, or setting a lower limit on certain dimensional parameters. Generally, a structural optimisation problem can be defined as:

$$\text{SO:} \begin{cases} \min\limits_{\boldsymbol{x}} \quad g_0(\boldsymbol{x}) \\ \\ \text{such that} \quad g_i(\boldsymbol{x}) < 0 \end{cases} \tag{2.1}$$

where g_0 is the objective function, $\boldsymbol{x}$ are the design variables and g_i are the constraints. The process of minimising g_0 is done in iterations where the design variables are updated. The process of updating the design variables is done using a convex approximation around the current values. Such methods are e.g. CONLIN [8] or MMA [27], where the latter is predominantly used. The sensitivity analysis is predominantly performed using either direct or adjoint differentiation, where the choice is made mainly depending on the amount of constraints of the optimisation problem.

A design iteration typically consists of the following main steps:

→ Using the current design variables, update the FE-formulation

→ Solve the FE-problem

→ Using the solution quantities, find the new values of g_0 and constraints g_i

→ Perform sensitivity analysis to find gradients g_0' and g_i'

→ Update design variables using new-found gradients

→ Check convergence

→ Repeat

2.2 Shape Parametrisation

When performing shape optimisation on a structure, there are many different ways of representing and controlling changes to the boundary. The *fictitious energy*-approach is one such way [22]. Being parameter-free, the fictitious energy approach can be advantageous to use. However, such parameter-free methods are less intuitive in regard of how control variables change the shape. On the other hand, methods that introduce additional parameters, such as splines, give a user-friendly, intuitive and adaptive way of representing shapes. Specifically, B-splines will be used for this thesis. They are a special case of the more general method NURBS (non-uniform rational B-spline) used in CAD-systems. Starting in one dimension, a curve can be represented by control points $\boldsymbol{V}_i$ and the basis functions $B_{i,p}$: [24]

$$\boldsymbol{X}(u, \boldsymbol{V}_i) = \sum_{i=1}^{n} B_i^p(u)\boldsymbol{V_i}. \tag{2.2}$$

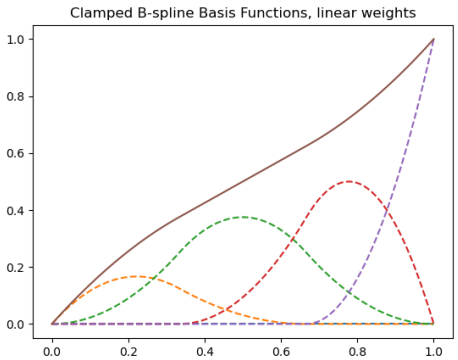
The basis functions $B_{i,p}$ are defined recursively using the knot vector $U = [u_0, ..., u_{p+n+1}]$:

$$B_i^p(u) = \frac{u - u_i}{u_{i+p} - u_i} B_i^{p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1}^{p-1}(u) \tag{2.3}$$

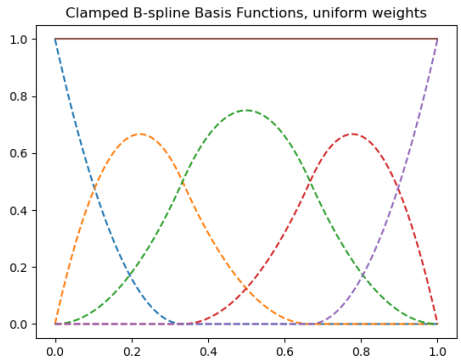
with the base case being $p = 0$:

$$B_i^0(u) = \begin{cases} 1 & \text{if} \quad u_i < u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} . \tag{2.4}$$

Due to the definition of these splines the knot vector must consist of non-decreasing values. Though, the vector can consist of multiple instances of the same value. In fact, having multiple instances of the same value is a vital technique for having the curve go through the first and last control point. Having, for example, three zeros at the start of the knot vector makes the first second degree basis function, i.e. $B_{0,2}$, clamped. This means its starting value is $\boldsymbol{V}_0$. Figures 2.1-2.2 shows the difference between clamped and non-clamped basis functions, and how they affect the final curve[24].

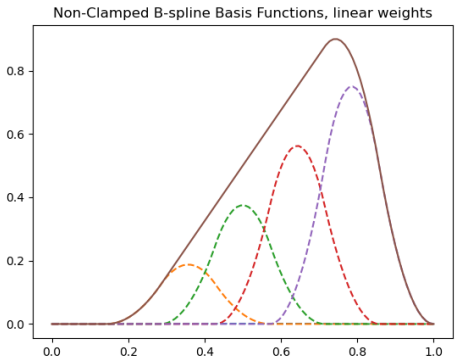


(a) Clamped basis function with linearly increasing weights

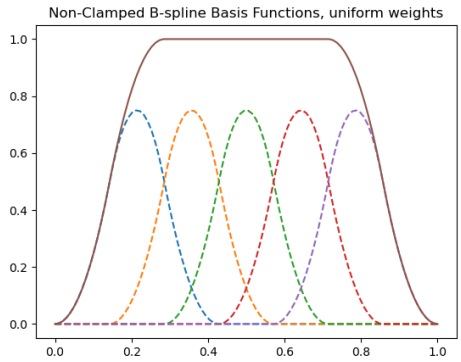


(b) Clamped basis function with uniform weight

Figure 2.1: Diagrams showcasing the second degree weighted basis functions as dotted lines, and the resulting curve as a solid line



(a) Non-clamped basis functions with linearly increasing weights



(b) Non-clamped basis functions with uniform weights

Figure 2.2: Diagrams showcasing the second degree weighted basis functions as dotted lines, and the resulting curve as a solid line

Figures 2.1-2.2 show that for the end points to pass through a certain, non-zero, point, the basis functions must be clamped. Using a combination of clamped and unclamped splines, an arbitrary curve representing the anvil surface, as shown in Figure 2.3, can be created.

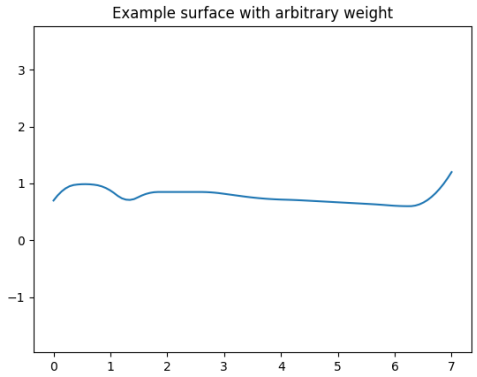


Figure 2.3: An arbitrary anvil parametrised by a b-spline.

2.3 Shape Optimisation

Shape optimisation is a form of structural optimisation, where the optimiser is limited to only changing the boundary of the given domain without changing the topological properties. Shape optimisation generally uses iterative approaches for solving the problem. This can be performed with gradient based methods or using surrogate models [31].

When optimising the boundary of an object, the shape of the surface needs to be controlled using design variables. This is typically accomplished through parameterisation. One may initially think parameterising the nodes directly is an adequate approach, but this method tends to create numerical difficulties due to jagged boundaries and inverted elements. Instead, splines provide an intuitive and functional way of parameterising the boundary [24]. If equation 2.2 is furthered by

defining:

$$\boldsymbol{V}_i = \boldsymbol{V}_i^0 + \begin{bmatrix} \alpha_i^h L_x \\ \alpha_i^v L_y \end{bmatrix}, \tag{2.5}$$

the α may be used as design variables in the optimisation. This yields an easy to control surface with an arbitrary, but controllable, level of differentiability.

2.3.1 Challenges with Gradient Based Methods

Sensitivity analysis is both the most vital and one the most challenging parts of structural optimisation. Without proper gradients, gradient based methods fail to update the design variables correctly. When the objective function becomes more complex, and the underlying physics and material models become more advanced, the sensitivity analysis in turn also becomes more complex. As for any optimisation with contact, the sensitivity analysis becomes significantly more problematic. The challenge presented by contact mechanics is the non-linear and non-smooth nature of the sensitivities, according to Sjövall and Wallin [25] (see [24] for a more thorough explanation). A change in design can suddenly introduce contact, and thus a force, which is non-trivial to account for in the optimisation. Since the scope of this thesis includes optimising with contact present, gradient based optimisation will be a challenge.

Further, since the material model for the paperboard is advanced, containing dozens of internal variables, the derivatives not related to the contact phenomenon will also be challenging. Differentiating the stiffness matrix with respect to the design variables in turn, results in one derivative of each of the internal variables with respect to the design variables. To exemplify: assume that the stiffness matrix has a dependence on plastic strains ϵ_p and damage κ, both quantities relevant for paperboard. Then, the derivative of the stiffness matrix will be:

$$\frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial x_i} = \frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial \boldsymbol{\epsilon}_p}\frac{\partial \boldsymbol{\epsilon}_p}{\partial x_i} + \frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial \boldsymbol{\kappa}}\frac{\partial \boldsymbol{\kappa}}{\partial x_i} \tag{2.6}$$

For further explaination of sensitivity analysis, see appendix A. Similar to the displacements, each of these derivatives with respect to the design variables will in turn require solving a non-trivial adjoint equation. In the context of developing a methodology for Tetra Pak, to simplify the optimisation workflow, it becomes unviable to require the engineer to perform the sensitivity analysis in this manner.

Due to the reasons presented, gradient based optimisation becomes unfeasible and utilising surrogate models for the design updates becomes the clear choice. The method can be likened to performing regression on measured data and using the regression model as a substitute, or surrogate, for the gradients in the sensitivity analysis. Various surrogate models and uses are described in further detail in section 3.3.

Chapter 3

Bayesian Optimisation

3.1 Introduction to Bayesian Optimisation

Bayesian optimisation (BO) is a strategy for optimising expensive black-box functions using machine learning [10][30]. As a versatile strategy, it has the capability to significantly reduce energy consumption, computational cost and capture uncertainty in the target function. When the unknown evaluated function lacks a functional form and is expensive to evaluate BO is particularly effective [10]. BO utilises a surrogate to model the unknown function, but unlike other surrogate methods it distinguishes itself by being based on Bayesian statistics and using acquisition functions to decide where to evaluate the objective next.

Typically BO is implemented as a sequential design strategy, meaning that all data points are added sequentially. However, Greif et al. [10] suggests that properly implementing an initial sampling method can significantly increase the accuracy of the model, while using fewer data points. After the initial sampling, an acquisition function will be used to determine where the next data point will be. There exists a myriad of surrogate models, initial sampling methods, and acquisition functions, a few of which will be mentioned, detailed or explained in this thesis [10].

3.2 Design of Experiments

When constructing a model of the relationship between variables and their corresponding function value, the goal is to perform a set of computations that capture as much information about the underlying process as possible. Typically, this can be achieved by discretising the variables and performing a computation for every possible permutation of values. When the number of variables are few, this is feasible, as e.g. one or two variables discretised with 5 values correspond to 5 or 25 computations respectively. When increasing the design space to more dimensions however, this rapidly becomes unfeasible due to the exponential growth, with e.g. 10 variables using the same discretisation requiring 5^{10} computations to capture every permutation. In order to reduce the number of computations but retain the ability to capture the dynamics of the variables, a Design of Experiment (DOE) can be performed.

The general idea of a DOE is to, by carefully choosing variable combinations, capture the general system dynamics while keeping the number of sampling points low [10][18][20]. A well executed initial sampling method improves the surrogate model's accuracy and reduces the number of evaluations, leading to a more efficient optimisation process. It is important to consider the dimensionality of the sampling space and the sampling budget when choosing a method. A sampling method may work well in few dimensions, but if it does not scale with dimensionality it will eventually perform poorly. Depending on prior knowledge and dimensionality of the situation at hand, different methods are preferred. When no knowledge exists at all, space-filling methods are ideal

as they allow the surrogate model to gain information about all different regions of the sample space, making sure no critical region is omitted. Low-discrepancy quasi-random sequences are able to uniformly distribute sample points across the entire sample space, making them well suited for this task [6][10]. There exist several methods and algorithms based on statistical measures for constructing a DOE, among which Latin Hypercube Sampling (LHS), Hammersley sequence sampling (HSS), and Sobol Sequence Sampling (SSS) will be described in further detail.

Before mathematically defining the methods, we will introduce them with an analogy. Consider a cinema about to hold a screening attended by 40 individuals. If we were to distribute them with a latin hypercube we start by filling the auditorium with 40 rows and 40 columns of seats. We then ask the attendants to randomly sit down in a seat, instructing them to make sure no one else is sitting in the same row or column. If, instead, a Hammersley sequence were to be used, we fill the auditorium with 40 seats, all placed such that there is as much space around them as possible. We then ask the attendants to sit down in whatever seat they would like to. Lastly, we fill the auditorium with a Sobol sequence. We hand the first attendant a seat and ask them to sit down randomly in the auditorium, then ask the next attendant to sit down in a spot that is as far away from the other attendant and the walls as possible. All following attendants are then asked to sit down as far away from all, already sitting, attendants and the walls as possible, until the cinema is filled. If more people show up, they too can be given a chair and the same instructions. In fact, with a Sobol sequence there is no limit to the number of attendants.

In reality, none of these methods of distributing attendants would be particularly popular. In fact, the Sobol method would probably result in angry people wondering why they have to carry their heavy seat to a position far away from their friends. Fortunately, sample points have no feelings.

3.2.1 Latin Hypercube Sampling

First introduced by McKay et al. [18] Latin Hypercube Sampling is a sampling method used for minimising the amount of sample points, $\boldsymbol{X}_k$ needed, while attempting to sufficiently cover the available sample space S [18]. Today it is a popular sampling method that many have tried to improve or alter to optimise certain qualities of the method. Komeilizadeh et al. [14], Leary et al. [17], Renardy et al. [21], and Shang [23] all present, or mention, different implementations of LHS optimised for different purposes. LHS is significantly more expensive and less space-filling than established low-discrepancy quasi-random method [6][21]. Since the purpose of this report is not to determine which LHS-alteration performs best, we have chosen to limit our research to the simple LHS method and use it comparatively to the low-discrepancy quasi-random methods.

Mathematical Definition

LHS builds upon stratified sampling, which in turn builds upon random sampling. For random sampling, let $\boldsymbol{X_1, ..., X_N}$ be randomly chosen sample points from $F(\boldsymbol{x})$, where $F(\boldsymbol{x})$ is a known probability distribution function for $\boldsymbol{x} \in S$. An issue that arises with this method is that it does not guarantee that the entire sample space, S, is represented in the statistical model. Although stratified sampling will not be explained in detail, an important idea of the model is to address this issue by partitioning the sample space S into I disjoint strata S_i [18].

By dividing each of the input variables X_k into N strata (see figure 3.1, where each dimension is divided into 6 strata), and sampling once from each stratum we ensure that all portions of the distribution are represented by input values. Although the probability distribution may be known, it is here set to a uniform distribution, making each point equally possible. Let each sample from the different strata of the different input variables X_k be $X_{kj}, j = 1, ..., N$. These values then form the sample points for the X_k-component in $\boldsymbol{X}_j, j = 1, ..., N$ [18].

The previous paragraph describes the selection process as described by McKay et al. [18]. Stein [26] offers a different explanation based on McKay's. Assuming that F is specified, it offers the joint distribution of the random vector of parameters $\boldsymbol{X}_j$. Now F_k denotes the cumulative distribution of X_k, and let X_{jk} be the kth component of $\boldsymbol{X_j}$. We define $P = (p_{jk})$ to be an $N \times K$ matrix, with columns of random permutations of {1,2,...,N}. We also define $\xi_{jk}(j = 1, ..., N; k = 1, ..., K)$ to be NK independent identically distributed U[0,1] random variables independent of P. X_{jk} can then be defined by: $X_{jk} = F_k^{-1}(N^{-1}(p_{jk} - 1 + \xi_{jk}))$ [26].

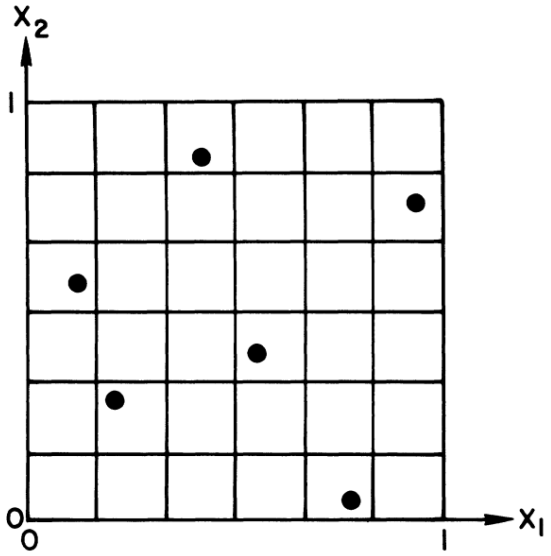


Figure 3.1: A latin hypercube DOE with N = 6, K = 2 for $\boldsymbol{X} = (X_1, X_2)$ distributed uniformly [26].

As described by McKay et al. the hypercube in figure 3.1 has only a point in each column and row. The more mathematically focused description by Stein provides additional insight into the formation of the hypercube. The p_{jk} determines in which cell the sample point $\boldsymbol{X}_j$ is located, while ξ_{jk} determines where in the cell X_j is located.

Figure 3.1 shows how a hypercube can be constructed in 2D. When the amount of design variables $K > 2$ it is no longer possible to visualise the cube in two dimensions, we therefore transition into vector notation: $\boldsymbol{X}_j = (X_{j1}, X_{j2})$. Exemplifying with $N = 2, K = 2$: $\boldsymbol{X}_1 = (X_{11}, X_{12}), \boldsymbol{X}_2 = (X_{21}, X_{22})$. Should we increase the amount of variables to $K = 3$ the number of variables exceed the number of samples. However, since each coordinate is only stratified into two different strata, it is still possible to cover all portions of the cube, despite having fewer samples than variables: $\boldsymbol{X}_1 = (X_{11}, X_{12}, X_{13}), \boldsymbol{X}_2 = (X_{21}, X_{22}, X_{23})$. While possible to create a latin hypercube this way, it is obvious that having fewer samples than variables makes the system underdetermined.

Implementation

For the sake of ease, while implementing the benchmarking, we elected not to have random sampling within the strata, but rather chose the strata's centre points as possible sample points. What follows is a short description of the algorithm for LHS using pseudo code.

Algorithm 1 LHS

for each design variable α **do**
  Initialise the possible values from lower to upper bound
  Shuffle them
  Add to DOE
end for

3.2.2 Hammersley Sequence Sampling

For systems with large factor ranges of the controls and noise the input-output relationships are complex. Higher complexity requires a greater number of samples for reliable approximations. An increase in sample size, means an increase in computational cost. Hammersley sequence sampling was introduced by Diwekar and Kalagnanam in 1997 as a means to optimise the sampling, requiring far fewer sampling than other sampling methods.

Mathematical Definition

The radix R of a numeral positional system is the number of digits used to represent numbers. For the binary system $R = 2$, only 0 and 1 are used, while $R = 10$ for the decimal system as the ten digits $0 \dots 9$ are used. Any integer n can be represented in radix-R notation as follows:

$$n = n_m n_{m-1} \cdots n_0 = n_0 R^0 + n_1 R^1 + \cdots + n_m R^m, m = \text{int}(\log_R(n)) \tag{3.1}$$

Reversing the order of n around the decimal point creates the so called *inverse radix number* $\phi_R(n)$.

$$\phi_R(n) = 0.n_0 n_1 \cdots n_m = n_0 R^{-1} + n_1 R^{-2} + n_m R^{-m-1} \tag{3.2}$$

Let us exemplify this with the number 18_{10}. Representing this number with radix 10 is trivial, it is simply 18. Instead we will use radix 2 to familiarise the reader with the process.

$$m = \text{int}(\log_2(18)) = \text{int}(4.09) = 4 \tag{3.3}$$

$m = 4$ means we will need five digits to represent the number, see equation 3.1. In radix 2 this becomes:

$$18_{10} = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 = 10010_2. \tag{3.4}$$

Reversing the order gives us:

$$\phi_2(18) = 0.01001_2 = 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5} = \frac{9}{32}. \tag{3.5}$$

We then define the Hammersley points on a k-dimensional cube from the following van der Corput-sequence:

$$\boldsymbol{z}_k(n) = \left(\frac{n}{N}, \phi_{R_1}(n), \cdots, \phi_{R_{k-1}}(n)\right) \quad \text{where } R_i \text{ is the } i\text{-th prime number.} \tag{3.6}$$

With the jth Hammersley point being $\boldsymbol{x}_j(n) = 1 - \boldsymbol{z}_{kj}(n)$

Example of Hammersley sequence generation

To exemplify HSS we will consider the following situation. We need to sample three points in three dimensions. While it is easy to extend this into four, or more, points or dimensions, an entirely new sequence has to be created. From the sampling description we require $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ with $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, x_{i3}\}$. Starting with generating the first point:

From the mathematical definition we remember that we first have to obtain the sequence z_{kj}. The first part of that sequence $\frac{n}{N} = \frac{1}{3}$. Computing $\phi_{R_1}(n)$ needs a few more steps.

$$\phi_{R_1}(1) = [n = 1_{10} = 1_2] = 0.1_2 = 0.5_{10} \tag{3.7}$$

Likewise:

$$\phi_{R_2}(1) = [n = 1_{10} = 1_3] = 0.1_3 = \left(\frac{1}{3}\right)_{10} \tag{3.8}$$

$\Rightarrow \boldsymbol{z}_{31} = \left(\frac{1}{3}, 0.5, \frac{1}{3}\right)$

The second point is then defined as:

$$\frac{n}{N} = \frac{2}{3} \tag{3.9}$$

$$\phi_{R_1}(2) = [n = 2_{10} = 10_2] = 0.01_2 = 0.25_{10} \tag{3.10}$$

$$\phi_{R_2}(2) = [n = 2_{10} = 2_3] = 0.2_3 = \left(\frac{2}{3}\right)_{10} \tag{3.11}$$

$\Rightarrow \boldsymbol{z}_{32} = \left(\frac{2}{3}, 0.25, \frac{2}{3}\right)$

Lastly, the third point:

$$\frac{n}{N} = \frac{3}{3} \tag{3.12}$$

$$\phi_{R_1}(3) = [n = 3_{10} = 11_2] = 0.11_2 = 0.75_{10} \tag{3.13}$$

$$\phi_{R_2}(3) = [n = 3_{10} = 10_3] = 0.01_3 = \left(\frac{1}{9}\right)_{10} \tag{3.14}$$

$\Rightarrow \boldsymbol{z}_{33} = \left(\frac{3}{3}, 0.75, \frac{1}{9}\right)$

Finally yielding the Hammersley sequence through $\boldsymbol{x}_j(n) = 1 - \boldsymbol{z}_{kj}(n)$:

$$\begin{cases} \boldsymbol{x}_1 = \left(\frac{2}{3}, 0.5, \frac{2}{3}\right) \\ \boldsymbol{x}_2 = \left(\frac{1}{3}, 0.75, \frac{1}{3}\right) \\ \boldsymbol{x}_3 = \left(0, 0.25, \frac{8}{9}\right) \end{cases} \tag{3.15}$$

3.2.3 Sobol Sequence Sampling

Similar to HSS, Sobol Sequence Sampling (SSS) is also a space-filling, quasi-random, low discrepancy sampling method. The key difference between the methods regards the construction of the sample. While Hammersley sampling is a non-extendable technique used for fixed-size sampling, i.e. where the amount of sample points must be specified a priori, the Sobol sequence is what qualifies as a sequential sampling method. The sequentiality of Sobol means that points can be added indefinitely while preserving the space filling quality of the method. This means that Sobol is better suited for sampling where the exact number of points is not known beforehand. Another similarity between HSS and SSS is that they are based on the representation of a base-10 number in a different base. Contrary to a base of primes, the foundation of the Sobol sequence is the uniqueness of the sequence generated through the binary representation of base-10 numbers. Examples of such representations are $6_{10} = 110_2$ and $1010101_2 = 85_{10}$, where each unique sequence of ones and zeros correspond to one and only one base-10 number. SSS is two magnitudes cheaper than LHS when computed in series, with the capability for parallelisation since, like HSS, it is a quasi-random sampling method [21]. What follows is a description of the mathematical implementation of a Sobol sequence.

Sobol Sequence Definition

The generation of a Sobol sequence is deterministic, reminiscent of HSS, but only given a specified *seed*. This means that for specific parameters, the Sobol sequence always produces the same sequence, in the same way that a random number generator produces the same sequence given the same seed. The seed for the Sobol sequence consists of a choice of *primitive polynomial* and initial *direction coefficients*. Both terms will be explained in further detail when necessary. To generate the i:th sample point for a k-dimensional design space,

$$\mathbf{x}_i = \{x_1, x_2, ..., x_k\}, \tag{3.16}$$

we begin by finding the binary representation of i (subscript 10 denotes base 10, subscript 2 denotes base 2):

$$i_{10} = (b_n \cdots b_3 b_2 b_1)_2 \tag{3.17}$$

where b_j is the j:th bit of the binary representation of i. These bits will become important in the end. Now, a choice of primitive polynomials must be made. One polynomial is constructed for each sample dimension (i.e. one for each design variable.). The form of the polynomials are:

$$p_d(x) = x^{o_d} + a_{1,d}x^{o_d-1} + a_{2,d}x^{o_d-2} + \cdots + a_{o_d-1,d}x + 1, \quad a_{p,d} \in \{0, 1\} \tag{3.18}$$

It is important to note that this polynomial is not to be evaluated, rather it acts as a way to store information, where the degree of the polynomial o_d and it's coefficients $a_{p,d}$ are important. As far as it regards the reader, the polynomial is simply a set of numbers. The choice of this set of numbers for each design variables acts as a seed and controls how the sequence is recursively generated. Standard choices exist for seeds [12]. The next important concept for the Sobol sequence are the so-called direction coefficients $m_{j,d}$. These are used to find the *direction numbers* $v_{j,d}$ whose use will be clarified together with the bits at the end. The direction coefficients are defined recursively using:

$$m_{j,d} = \left(\bigoplus_{l=1}^{o_d-1} \left(2^l a_{l,d} m_{j-l,d}\right) \right) \oplus (2^{o_d} m_{j-o_d,d}) \oplus m_{j-o_d,d} \tag{3.19}$$

Where $a_{l,d}$ are the coefficients from the primitive polynomial and the $\oplus$ operator is the bitwise XOR-operator, which works similar to an inverse Kronecker delta:

$$\begin{cases} 1 \oplus 1 = 0 \\ 0 \oplus 0 = 0 \\ 0 \oplus 1 = 1 \\ 1 \oplus 0 = 1, \end{cases} \tag{3.20}$$

The operator acts bitwise on each bit of the binary numbers, meaning $9_{10} \oplus 12_{10} = 1001_2 \oplus 1100_2 = 0101_2 = 101_2 = 5_{10}$. While the recursion appears complicated, the use is exemplified in a simple example in the following section. The use of the final parameter from the primitive polynomial o_d is to define how many initial values of direction coefficients need to be given. If a third degree primitive polynomial, i.e. $o_d = 3$, the initial three values $m_{1,d}, m_{2,d}, m_{3,d}$ have to be specified using e.g. a standard seed value. The direction coefficients are then used to find the direction numbers, which are defined as:

$$v_{j,d} = \frac{m_{j,d}}{2^j} \tag{3.21}$$

i.e. as binary fractions. This is done to generate all $v_{j,d}$ from $v_{1,d}$ through $v_{n,d}$. After finding all n direction numbers, we remember the initial bits from the binary representation of i, and the coordinate for design variable d in the i:th sample point is defined as:

$$x_d = b_1 v_{1,d} \oplus b_2 v_{2,d} \oplus \cdots \oplus b_n v_{n,d} \tag{3.22}$$

The coordinate is given in base 2 but can easily be transformed back to base 10, yielding a value between 0 and 1. This is then repeated for all design variables d from 2 through k, and the i:th sample point is created. The attentive reader might notice that this was specified for dimensions 2 through k. The first dimension follows a different rule-set and is independent of seed. For the first dimension, sample values are calculated using the van der Corput sequence in base 2. This simplifies to

$$m_{j,1} = 1 \implies v_{j,1} = \frac{1}{2^j} \tag{3.23}$$

and the samples first coordinate is simply:

$$x_1 = b_1\frac{1}{2} \oplus b_2\frac{1}{2^2} \oplus \cdots \oplus b_n\frac{1}{2^n} \tag{3.24}$$

Example of Sobol Sample Generation

To exemplify the generation of Sobol sequences, we can attempt to generate the e.g. sixth sample of a 3D design space. The choice of sample 6 is to show how multi-bit samples behave without too much work. We begin by finding the binary representation of $i = 6$:

$$6_{10} = 110_2 \implies \begin{cases} b_3 = 1 \\ b_2 = 1 \\ b_1 = 0 \end{cases} \tag{3.25}$$

From this, we see that

$$x_d = 0 \cdot v_{1,d} \oplus 1 \cdot v_{2,d} \oplus 1 \cdot v_{3,d} = v_{2,d} \oplus v_{3,d}, \tag{3.26}$$

meaning only the second and third direction numbers are used. Using table values from [12], we see that for the three dimensions, the seeds are:

d (dimension)	o_d	a (binary)	$m_{j,d}$
1	-	-	-
2	1	0	1
3	2	1	1,1

Table 3.1: Seed data for three dimensional Sobol sequence sample. Dimension 1 left empty, follows van der Corpus sequence.

From this, we can construct our 6:th sample point. We begin with the first coordinate:

$$x_1 = v_{2,1} \oplus v_{3,1} = \frac{1}{2^2} \oplus \frac{1}{2^3} = 0.01_2 \oplus 0.001_2 = 0.011_2 = 0.375_{10} \tag{3.27}$$

For the second coordinate, we begin by analysing the seed. $o_d = 1$ means we have our primitive polynomial as $p(x) = x + 1$, and $a = 0$ is redundant information, as there are no coefficients present. As the order o_d is one, we only need to initialise one direction coefficient, $m_{1,2} = 1$. To find the remaining direction coefficients, we iterate:

$$\begin{cases} m_{2,2} = 2a_1 m_{1,2} \oplus 2m_{1,2} \oplus m_{1,2} = 10_2 \oplus 01_2 = 11_2 = 3_{10} \\ m_{3,2} = 2a_1 m_{2,2} \oplus 2m_{2,2} \oplus m_{2,2} = 110_2 \oplus 11_2 = 101_2 = 5_{10} \end{cases} \tag{3.28}$$

and further:

$$\begin{cases} v_{1,2} = \frac{m_{1,2}}{2} = 0.1_2 \\ v_{2,2} = \frac{m_{2,2}}{2^2} = 0.11_2 \\ v_{3,2} = \frac{m_{3,2}}{2^3} = 0.101_2 \end{cases} \tag{3.29}$$

$$x_2 = v_{2,2} \oplus v_{3,2} = 0.11_2 \oplus 0.101_2 = 0.011_2 = 0.375_{10} \tag{3.30}$$

Now, the coordinate for the third dimension can be found. The table value $o_d = 2$ states that $p(x) = x^2 + a_1 x + 1$. The table value for $a = 1_2$ means $a_1 = 1$, i.e. $p(x) = x^2 + x + 1$. While it does not appear obvious here, the table value for a is to be interpreted as a binary number, i.e. if the table states $a = 9$ it means $a = 1001_2$, meaning $a_4 = 1, a_3 = 0, a_2 = 0, a_1 = 1$. Returning to dimension 3, the table states that $m_{1,3} = m_{2,3} = 1$. To find the final direction coefficient, we use the recursion:

$$m_{3,3} = 2a_1 m_{2,3} \oplus 2^2 m_{1,3} \oplus m_{1,3} = 10_2 \oplus 10_2 \oplus 1_2 = 00_2 \oplus 01_2 = 1_2 = 1_{10} \tag{3.31}$$

and thus:

$$\begin{cases} v_{1,3} = \frac{m_{1,3}}{2} = 0.1_2 \\ v_{2,3} = \frac{m_{2,3}}{2^2} = 0.01_2 \\ v_{3,3} = \frac{m_{3,3}}{2^3} = 0.001_2 \end{cases} \tag{3.32}$$

We can now find the third coordinate,

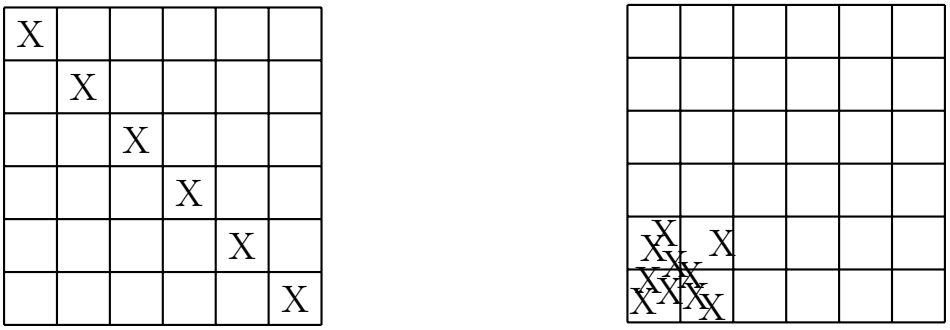
$$x_3 = v_{2,3} \oplus v_{3,3} = 0.01_2 \oplus 0.001_2 = 0.011_2 = 0.375_{10} \tag{3.33}$$

Finally, the sixth sample point in a 3D design space is:

$$\mathbf{x}_6 = (0.375, 0.375, 0.375) \tag{3.34}$$

3.2.4 Comparing the Sampling Methods

LHS has some major advantages over random sampling, one being that it guarantees that values are sampled from each strata, exactly once. While this means that no two samples will be identical, and that each portion of the individual dimensions will be sampled, there is no guarantee that the distribution will be adequate for interpolation over the entire sample space.



(a) Latin hypercube sampling

(b) Pure random sampling

Figure 3.2: Examples of possible Latin hypercube sampling and pure random sampling.

Figure 3.2 shows suboptimal samples from random sampling and Latin Hypercube sampling. While the LHS sample covers all portions of the individual axes, it does not sufficiently cover the domain they span. The LHS sample will not show which variable is the contributing one. While this is the worst case possible for LHS, it can never guarantee that sampling will adequately cover the plane. The example of random sampling is worse than that of LHS, it covers neither the domain, nor the individual axes, and is therefore not an adequate sampling of the sample space.

There are different measures to quantify how space-filling an algorithm is. Shang [23] presents the following two commonly used measures: minimum pairwise distance, and maximum hole size. If we denote by $S_n = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ a sample with n sample points, with $\boldsymbol{x}_i \in \mathbb{R}^q$ and q being the number of design variables, then we can define the quantities as follows:

minimum pairwise distance:

$$d_{min}(S_n) \triangleq \min_{\substack{i,j \in \{1,...,n\} \\ i<j}} d(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{3.35}$$

maximum hole size:

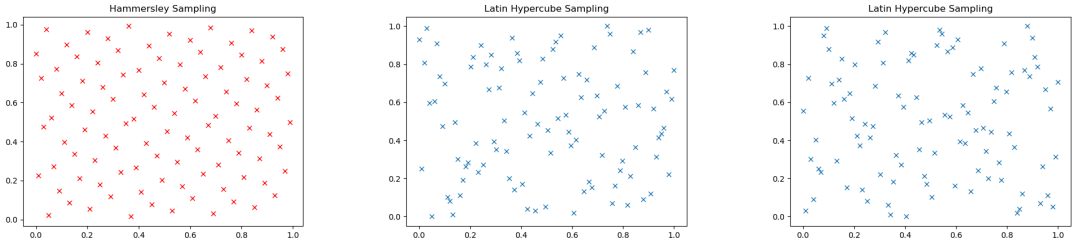
$$h_{max}(S_n) \triangleq \max_{\boldsymbol{x} \in [0,1]^q} \min_{i \in \{1,...,n\}} d(\boldsymbol{x}_i, \boldsymbol{x}). \tag{3.36}$$

For computational reasons the maximum hole size is an unpractical quantity to obtain. Instead, minimum pairwise distance is preferred as a measure of how space-filling a method [23]. While problematic to compute the maximum hole size, figure 3.3 shows LHS is less space-filling than HSS using these quantities.

In few dimensions, HSS has some advantages over LHS: it yields a more space-filling sample and can achieve a similar performance with fewer sample points [6]. However, in higher dimensions (e.g beyond 10) the performance degrades substantially [16]. Like HSS, SSS is also a low-discrepancy quasi-random sampling method; however, unlike HSS, it retains its efficiency much longer.

At this point different articles present somewhat conflicting information. Kucherenko et al. [15] presents HSS as a sampling method that requires up to 40 times fewer sampling for convergence than LHS, but does not endorse or recommend it as a method. Instead, the article claims that Sobol is the better sampling method overall, with superior performance to LHS at 20 dimension, and similar performance at around and above 40 dimensions. Only in very specific cases, and in low dimensions, does LHS outperform Sobol according to Kucherenko et al. [15].

In contrast, Dige and Diwekar [5] conclude that HSS is the superior method up to 40 dimensions, after which Sobol is the superior up to 100 dimensions. Beyond 100 dimensions, their own combined LHS-Sobol-method is proposed as a superior method to the previously mentioned [5].



(a) The results from Hammersley sequence sampling, identical each time.

(b) The results from an iteration of Latin Hypercube Sampling, random each time

(c) The results from an iteration of Latin Hypercube Sampling, random each time

Figure 3.3: The sample points from two runs of LHS, and HSS

3.3 Surrogate Modelling

When optimising using gradient based approaches, the gradients of the objective function are key in updating the design variables, and improving the objective function. When the connection between the objective function and the design variables is complex, as in the case for this thesis, surrogate modelling can be an efficient alternative to gradient based approaches. The principle of surrogate modelling is to impose a functional form on the objective function. This functional form may be e.g polynomial, logarithmic, or a Gaussian process. Then, an interpolation or regression can be performed to create a multidimensional surface, on which the optimisation can be conducted. Here, two methods for surrogate modelling are described, polynomial regression and Kriging modelling. To perform Bayesian optimisation, a Gaussian process needs to be used, where only the latter surrogate model is such a process. Although not able to be used in Bayesian optimisation, polynomial regression is a simple surrogate model and can easily be implemented to create an optimisation methodology, and is therefore included.

3.3.1 Polynomial Regression

An intuitive way to formulate a model for data points is to simply use a polynomial. Polynomials of the first and second order are the most simple, and a polynomial fit can be performed using the least squares method. A first degree polynomial can be used to approximate a function $f(\mathbf{x})$ in K design variables as:

$$\tilde{f}(\mathbf{x}) = \beta_0 + \sum_{k=1}^{K} \beta_k x_k, \tag{3.37}$$

where β_k are the polynomial coefficients. The tilde signifies that it is only an approximation of $f(\mathbf{x})$. A second degree polynomial offers more freedom in its formulation, as it can be limited to pure squares or also include square interaction terms between variables. For a purely square polynomial, it can be formulated as:

$$\tilde{f}(\mathbf{x}) = \beta_0 + \sum_{i=1}^{K} \beta_i x_i + \sum_{j=1}^{K} \beta_j x_j^2, \tag{3.38}$$

whereas a second degree polynomial that includes interaction terms can be written as:

$$\tilde{f}(\mathbf{x}) = \beta_0 + \sum_{i=1}^{K} \beta_i x_i + \sum_{j=1}^{K} \sum_{k \leq j}^{K} \beta_{jk} x_j x_k, \tag{3.39}$$

All these explicit expressions for $f(\mathbf{x})$ can be expressed more general as a linear equation on the form:

$$f(\mathbf{X}) = \boldsymbol{\beta}\mathbf{X} + \epsilon, \tag{3.40}$$

where ϵ is the error between the polynomial regression and the actual function value. Here, $\mathbf{X}$ gathers the terms of of the sums in a so called basis matrix, which has the form (parenthesised number denotes sample number 1 through N):

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_K^{(1)} & (x_1^{(1)})^2 & (x_2^{(1)})^2 & \cdots & (x_K^{(1)})^2 & x_2^{(1)}x_1^{(1)} & \cdots & x_K^{(1)}x_{K-1}^{(1)} & \cdots & x_K^{(1)}x_1^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_K^{(2)} & (x_1^{(2)})^2 & (x_2^{(2)})^2 & \cdots & (x_K^{(2)})^2 & x_2^{(2)}x_1^{(2)} & \cdots & x_K^{(2)}x_{K-1}^{(2)} & \cdots & x_K^{(2)}x_1^{(2)} \\ & & & & & & & & & \vdots & & & & \\ 1 & x_1^{(N)} & x_2^{(N)} & \cdots & x_K^{(N)} & (x_1^{(N)})^2 & (x_2^{(N)})^2 & \cdots & (x_K^{(N)})^2 & x_2^{(N)}x_1^{(N)} & \cdots & x_K^{(N)}x_{K-1}^{(N)} & \cdots & x_K^{(N)}x_1^{(N)} \end{bmatrix}$$

To perform the polynomial regression, the coefficients are found in the least squares sense, resulting in the model being:

$$\tilde{f}(\mathbf{X}) = \hat{\boldsymbol{\beta}}\mathbf{X}, \tag{3.41}$$

where coefficients are found using the equation with measured data $\mathbf{y}$ of $f(\mathbf{X})$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3.42}$$

The PRG model can be extended to higher degrees of polynomial, but the amount of basis functions needed to capture all interaction terms, and in response the amount of samples needed, increases rapidly. As a result of this, degrees higher than two become impractical to use, especially as the second degree already offers a convex approximation. A negative aspect of polynomial regression in terms of FE-simulations is that it is a regression, not an interpolation, i.e. when the approximation is not sufficiently valid, the simulation results are not correctly fitted to, and the ground truth of the results is lost.

3.3.2 Kriging Modelling

Another choice of surrogate model is Kriging modelling, named after a South African mining engineer, Danie Gerhardus Krige. Kriging modelling was initially created to predict ore distributions using systematic drill samples. This section will briefly explain Kriging modelling, with simplified notation. For a more exhaustive explanation, see [32]. The Kriging model, or Gaussian Process (GP) is a stochastic process, in which the error of the process has high correlation between points in near vicinity to each other, and low correlation for points far apart. In explicit terms, the general model can be written as:

$$\hat{y}(\mathbf{x}) = \hat{f}(\mathbf{x}) + \mathbf{r}(\mathbf{x})^T\mathbf{R}(\mathbf{y} - \mathbf{1}\hat{f}(\mathbf{x})), \tag{3.43}$$

where $\hat{y}(\mathbf{x})$ is the Kriging approximation of the value for a given point $\mathbf{x}$ in the design space, also called a *query point*. Both $\mathbf{r}(\mathbf{x})$ and $\mathbf{R}$ are correlation factors, where $\mathbf{R}$ is the correlation matrix, storing all correlation values between all sample points. The $\mathbf{r}(\mathbf{x})$ is vector containing the correlation values between the query point and all sample points. $\mathbf{y}$ is a vector of all sample responses, in this case the objective function values. Regarding the last term, $\hat{f}(\mathbf{x})$, there exists three different variations of how it is defined and they each give rise to a different Kriging model, which are explained in subsequent sections.

Regarding the aforementioned correlation, it is a key element of the Kriging model. Since errors are assumed to have higher correlation between points close in the space compared to points further apart, this need to be quantified in an explicit way. This is done by selecting a correlation function dependent on distance. For a point $\mathbf{x}_i$ the error is $\epsilon(\mathbf{x}_i)$ and subsequently, we define the components of the two correlation quantities as:

$$\begin{cases} \mathbf{R}_{ij} = \text{Corr}\left[\epsilon(\mathbf{x}_i), \epsilon(\mathbf{x}_j)\right] \\ \\ \mathbf{r}_i(\mathbf{x}) = \text{Corr}\left[\epsilon(\mathbf{x}_i), \epsilon(\mathbf{x})\right] \end{cases} \tag{3.44}$$

There exists several choices for the correlation function, all with different properties. Since the model will be used for optimisation, a differentiable correlation function is preferred, and one such choice is the family of Matérn kernels:

$$\text{Corr}[\epsilon(x), \epsilon(x')] = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\,\|x - x'\|}{\theta}\right)^{\nu+1} K_\nu, \tag{3.45}$$

where Γ is the gamma-function, K_ν is a modified Bessel-function. The range decay parameter, θ, acts as an importance factor, weighting certain samples as more important than others. This can

be likened to a radius of influence for a sample point. All the reader needs to concern themselves with at this point is that we set $\nu = p+1/2$, where we choose p as the number of times the Kriging model shall be differentiable, i.e we determine how smooth the approximation shall be [19].

Simple Kriging

Simple Kriging (SK) assumes that the underlying process has zero mean. In this case, $\hat{f}(\mathbf{x})$ is assumed to be zero, and thus allows the model to be written as:

$$\hat{y}(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{R}\mathbf{y}. \tag{3.46}$$

Ordinary Kriging

Ordinary Kriging (OK) also assumes that the underlying process has a static, but non-zero, mean μ. Finding the mean is simple using the sample data. The model then becomes:

$$\hat{y}(\mathbf{x}) = \mu + \mathbf{r}(\mathbf{x})^T \mathbf{R}(\mathbf{y} - \mathbf{1}\mu). \tag{3.47}$$

Universal Kriging

Universal Kriging (UK) is the most sophisticated Kriging model of the three included. Here, f is assumed to follow a certain trend, e.g. a polynomial. Therefore, this model can be seen as a more sophisticated alternative to PRG [19].

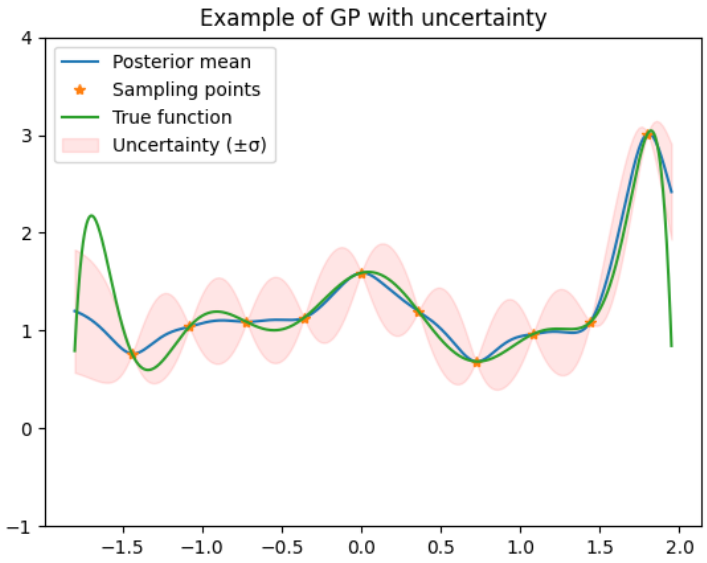


Figure 3.4: The figure show how the model and its confidence bounds differs from the true function.

A GP builds its approximations based on acquired points. Between the acquired points it uses interpolation to estimate the function value. This value is obtained through Bayesian statistics, meaning there is an interval in which it may lie. The model's function value is chosen as the mean of the interval, i.e the posterior mean of the approximation. The uncertainty is low close to the acquired points, and increases with distance. Therefore, the posterior mean is typically accompanied by the uncertainty when plotting the approximation, see figure 3.4.

3.4 Acquisition Functions

Improving the surrogate model requires evaluating additional points in the design space. Evaluating these points are generally computationally expensive in the context of FE-simulations, and randomly selecting points for evaluation leads to uninteresting areas being explored. Since this is a waste of computational resources, Jones et al. [13] introduced 'figures of merit' to determine where to evaluate next. Today these 'figures of merit' are known as acquisition functions and should reflect how much is to be gained from evaluating a certain point. An effective acquisition function needs to have two important properties. The first being that it values what is sought, while balancing *exploitation*, evaluating points in promising areas, with *exploration*, evaluating points in unknown areas. Since this balance is problem-dependent, a range of acquisition functions has been proposed to achieve this, i.a *Expected Improvement* (EI), *Probability of Improvement* (PI), and *Upper Confidence Bound* (UCB) [1].

An acquisition function integrates the information provided by the surrogate model, and then ranks all the points in the domain on their potential usefulness. The optimal subsequent candidate is found by performing an optimisation of the acquisition function. Hence the second important quality of an acquisition function: it must be significantly cheaper to evaluate than the objective function [1]. Expected Improvement is the most commonly used acquisition function and will be detailed below. Due to issues arising when using EI, UCB and parallel EI (qEI) were explored as alternatives, and will therefore also be detailed below.

3.4.1 Expected Improvement

This acquisition function elects the subsequent point for evaluation based on the expected improvement of each point. Defining:

$$\boldsymbol{x}_{n+1} = \arg\max_{x}(\mathrm{EI}(x)) = \arg\max_{x}(\mathbb{E}_{\tilde{f}}\left[\max\left\{0, \mu(x) - f_{\max}\right\}\right]) \tag{3.48}$$

With:

$$\begin{aligned}\mathrm{EI}(x) &= \int \left[\mu(x) - f_{\max - \sigma(x)\xi}\right]_+ \phi(\xi)\, d\xi \\ &= \left[z = \frac{\mu(x) - f_{\max}}{\sigma(x)}\right] = \begin{cases} z\boldsymbol{\Phi}(z)\sigma(x) + \phi(z)\sigma(x) & \text{if } \sigma(x) > 0 \\ 0 \quad \text{if } \sigma(x) = 0 \end{cases}\end{aligned} \tag{3.49}$$

Where ϕ and $\boldsymbol{\Phi}$ are the standard normal probability density function and cumulative distribution function, $\mu(x)$ is the posterior mean at x and f_{max} is the greatest observed value. EI can be altered to encourage exploration by adding a penalty term to z:

$$z = \frac{\mu(x) - f_{\max}}{\sigma(x)} - \frac{\epsilon}{\sigma(x)} \tag{3.50}$$

A low uncertainty $\sigma(x)$ makes the penalty parameter large, encouraging the function to explore uncertain regions [1][3]. While popular, Expected Improvement suffers from two serious issues: it is numerically unstable and can easily get stuck in a local extremum. To fix the first issue a logarithmic and stable variant of the function will be used in the implementation. To avoid getting stuck in local extrema UCB and qEI were tried as alternative acquisition functions.

3.4.2 Upper Confidence Bound

UCB is another acquisition function that has a controllable rate of exploration. Unlike EI its definition is straightforward, and its exploration parameter is very intuitive:

$$\boldsymbol{x}_{n+1} = \arg\max \mathrm{UCB}(\boldsymbol{x}) \tag{3.51}$$

With:

$$\text{UCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \beta\sigma(\boldsymbol{x}) \tag{3.52}$$

The parameter β is simple to change and, intuitively, a high value of β creates an exploratory acquisition function - while a low value of β creates an exploitative acquisition function [1][3]. Although the function of β is intuitive, balancing it is not trivial.

3.4.3 Parallel Expected Improvement

The last acquisition function that was explored is qEI. It has an analytical expression, but not a closed one, and has to be computed using Monte Carlo sampling. Its mathematical definition is:

$$\text{qEI}(\boldsymbol{X}) = \mathbb{E}_{f(\boldsymbol{X})}\left[\max_{j=1,\dots,q}\{[f(\boldsymbol{x}_j) - f_{max}]_+\}\right] \tag{3.53}$$

The mathematical definition is shown here to convince the reader that the idea behind the function is similar to EI's idea. However, the expression above cannot directly be used to compute the expected improvement. Instead, numerical tricks have to be used. These tricks will not be shown, nor will it be explained how the computations work in closer detail in this report. For that, the reader is referred to Ament et al. [2], Ginbourger et al. [9] and Wang et al. [29]. Ament et al. also founded the logEI family of acquisition functions, these functions evaluate like regular EI, maintaining the same gradients, while being more numerically stable. Like qEI, these functions' mathematical definitions are complex, and including their definitions here would not provide any additional value to the report. Therefore, once again, the reader is referred to Ament el al. for the definitions and inner workings of those functions.

Although we will not analyse the mathematical expression further, it will be explained in short how the function finds its points. The batch of points is chosen such that they maximise the function: $\boldsymbol{X}_{n+1} = \arg\max \text{qEI}$. These points add more value to the function when they are not in close proximity, as there is little to be gained from evaluating right next to another point. This means that qEI is encouraged to not cluster the points; instead, it tends to spread them out in promising regions, thereby maximising the expected improvement of the batch. As this is done with Monte Carlo sampling the function may not return the same points every time for the same state of the model, adding an element of randomness to the acquisition.

3.4.4 Examples of Acquisitions

This section will show how some initial acquisitions to show the reader how the different acquisition functions evaluate the points. The function that will be used and the initial sampling is seen in figure 3.5.

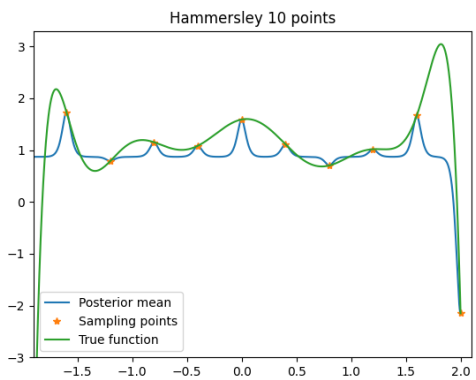


Figure 3.5: The true function and the initial sampling used

Expected Improvement

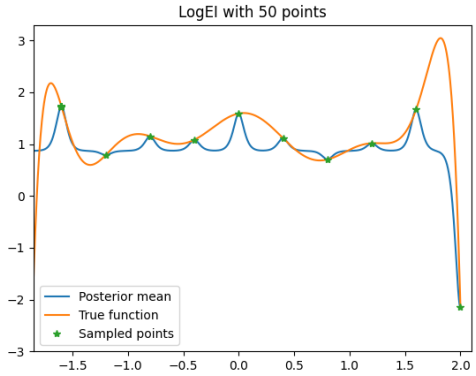


Figure 3.6: Expected Improvement with 50 additional points

Although it is not clear, there are 60 total points in the figure. The leftmost point is actually 51 points in almost the exact same position. If there would not have been artificial added noise of 10^{-4}, these points would in fact end up in the exact same position. When the initial sampling is poor the GP can get a "spikey" appearance. This initial fit causes the EI to be maximised in the maximum sample point, causing the acquisition function to elect an existing sample point for evaluation. This causes the GP to maintain the appearance throughout the acquisition, ensuring the subsequent points are the same. This phenomenon originally occurs due to bad hyperparameter (the internal parameters of the GP) fitting. EI proved unable to break out of this phenomenon, which is why UCB and qEI were tried instead. This does not necessarily make EI a worthless acquisition function, it can perform well when the surface is convex, or when the GP's hyperparameter are well fitted.

Upper Confidence Bound

For the following examples Upper Confidence Bound (UCB) has been used instead of UCB. UCB is used for minimising while UCB is used for maximising. We chose to maximise the example function in the image, and therefore used UCB instead of UCB. It is worth noting that the β in the following images is the square of the β in equation 3.52 due to the BoTorch implementation.

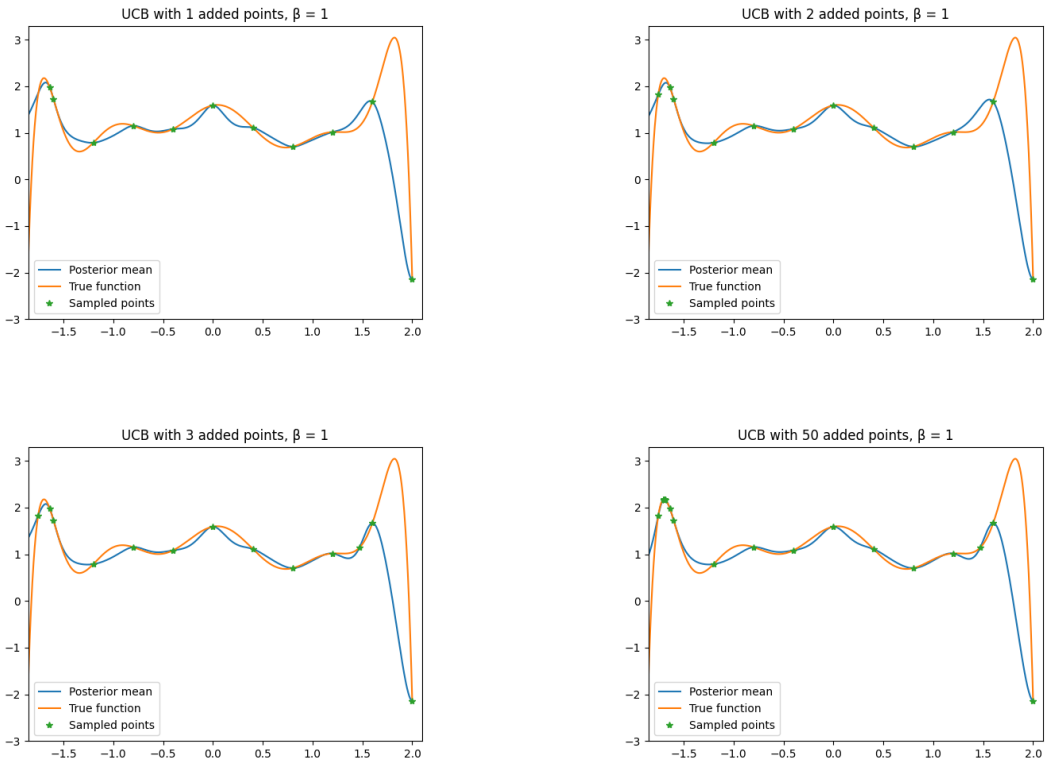


Figure 3.7: Figures with different amount of acquisitions for beta = 1

Having a low β, in this case 1, causes the function to be very exploitative. Immediately it starts sampling around the best initial point. This leads the model to a good approximation of the vicinity around the discovered maximum; however, the true maximum remains completely unexplored.

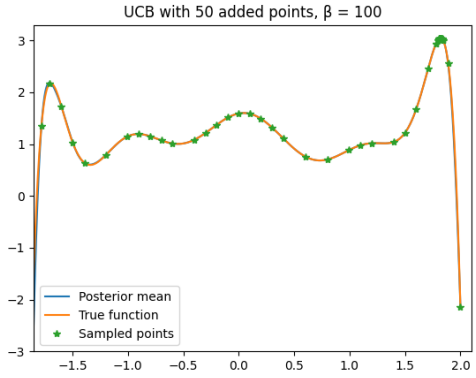


Figure 3.8: UCB with for $\beta = 100$ with 50 additional points

Having a higher β makes the function much more exploratory. In the beginning of the acquisition phase the samples are spread evenly, while the uncertainty remains high. When the uncertainty eventually becomes low, the function starts exploiting the most promising region, placing the remaining points there. For the example function, one could argue that the exploration coefficient is too high, as extensive exploration is unnecessary when the sole objective is to identify the

maximum. Since the complexity of the function is unknown a priori, selecting an appropriate value of to balance exploration and exploitation is challenging.

Parallel Expected Improvement

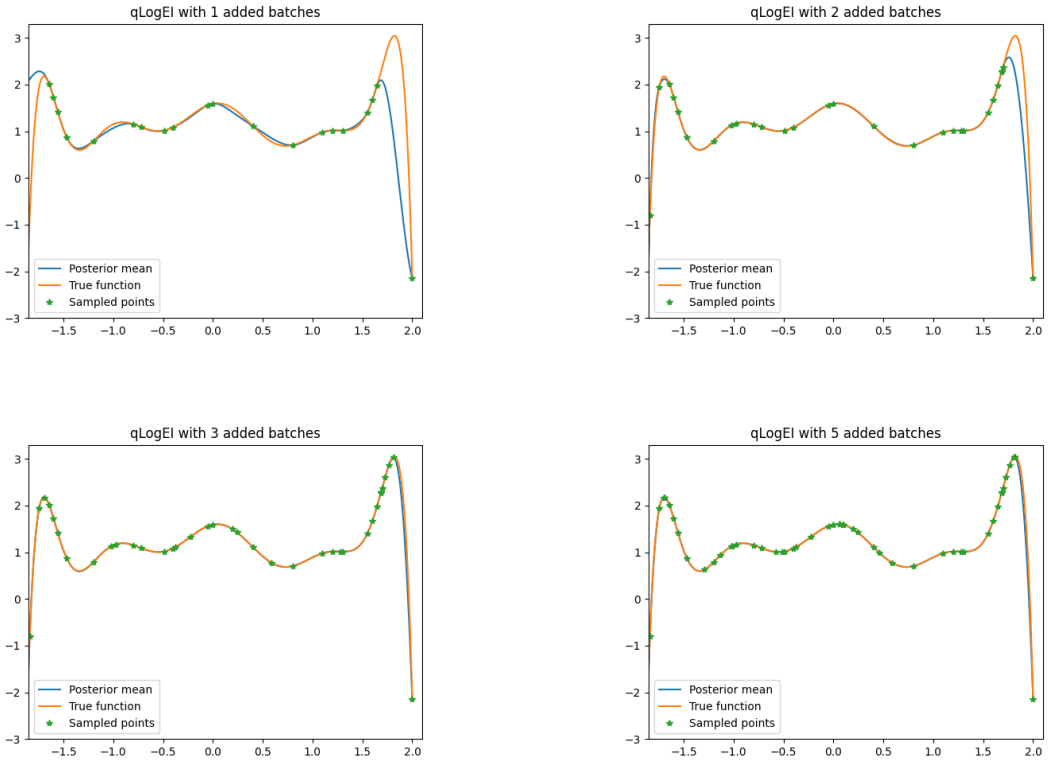


Figure 3.9: The different stages of acquisition with qEI, q = 10

qEI adds many more points at every step of acquisition than EI and UCB. Placing points at different promising regions, qEI ends up having a higher density of points in local maxima than in local minima. This yields a good approximation of the model in the interesting regions while maintaining a level of exploration.

Chapter 4

Benchmarking and Initial Testing

As all projects evolve, both the scope and methods are subject to change. The original scope consisted of starting with a benchmarking example and continuing with an attempt of optimisation of the sealing anvil using commercial software. The role of optimisation was to be treated as more of a design tool rather than the main research topic. While what became the new scope of the project is contained in the following chapters, these initial tests are included here, as they are still relevant to the subject. The first benchmarking test is an implementation of convex optimisation with a polynomial surrogate model, and can be seen as a simpler alternative to the Bayesian optimisation that was developed. The second part, dubbed "Initial Testing in HyperWorks", details the implementation of optimisation in HyperStudy and is relevant for future work at Tetra Pak. Each of the parts are presented in its own closed format with an introduction and description of the problem setup, the methodology, the results, or lack of, as well as a discussion.

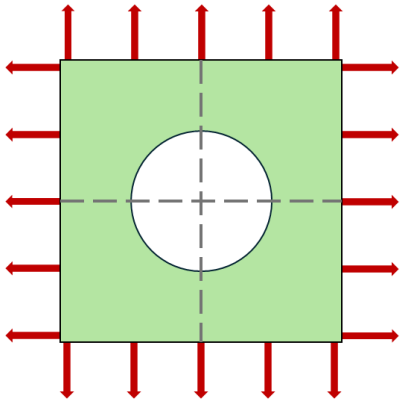
4.1 Benchmarking Test

As a first evaluation of using surrogate models for shape optimisation, a benchmarking test was performed. Although not related to the actual optimisation problem of the project scope in neither choice of optimisation method or problem formulation, the benchmarking still provided experience using sampling and surrogate models to replace sensitivity analysis. The benchmarking was performed in the early stages of the project when the methodology was still in the early stages of research and development, and is thus a methodology based on convex optimisation rather than Bayesian optimisation. An analysis of a convex optimisation methodology is still relevant though, as it is a viable methodology for problems where a convex approximation is valid. The following section provides further details about the benchmarking problem.

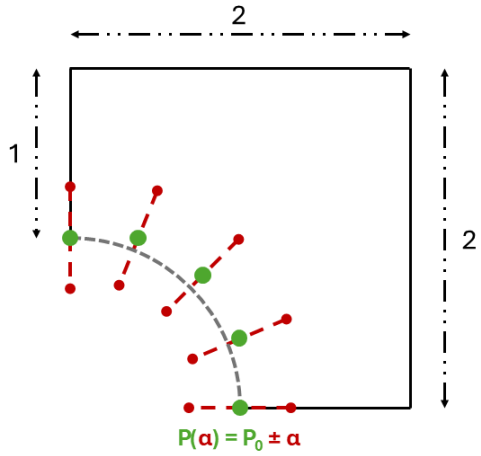
4.1.1 Problem Description

The benchmarking problem selected was shape optimisation of a hole in a square plate subject to biaxial traction. A sketch of the problem can be seen in Figure 4.1a, where the hole in the centre is parametrised using a spline with control points. Due to the biaxial symmetry present in the problem, only the first quadrant is chosen for computation as to limit dimensionality in design variables. This also reduces computational effort. The objective function for the optimisation is to minimise compliance of the plate with a volume constraint. A mathematical formulation of the optimisation problem can be seen in equation (4.1). The optimised configuration for the problem is with design variables resulting in a circular hole of the plate [28].

$$\text{SO:} \begin{cases} \min\limits_{\mathbf{x}} \;\; g_0(\mathbf{x}) = \mathbf{F}^T\mathbf{u} \\ \\ \text{such that} \;\; V(\mathbf{x}) < V_{\mathbf{max}} \end{cases} \tag{4.1}$$



(a) Illustration of the problem setup. Red arrows represent the applied traction. Dashed grey lines represent the two symmetry lines in the problem.



(b) Example of the geometry of the plate including control points in green and their respective constraints in dashed red lines. The spline is represented using a grey dashed line.

Figure 4.1

4.1.2 Surrogate Based Optimisation Algorithm

A simple method for optimisation using surrogate modelling is the surrogate based optimisation (SBAO) method [20]. The method has a quite simple algorithm, as explained in algorithm 2. This method acts in a similar way to the expected improvement acquisition function, in that it finds the extremum of the surrogate model and chooses that point for evaluation. SBAO is used in the benchmarking test as the optimisation method after sampling.

Algorithm 2 SBAO

Require: Design variable sample matrix $\boldsymbol{A}$, objective function values $\mathbf{g}_0$
  while res $>$ tol **do**
    Create surrogate model $\hat{f}(\boldsymbol{\alpha})$ using $\boldsymbol{A}$ and $\mathbf{g}_0$
    Minimise to find $\boldsymbol{\alpha}_{\min}$: $\min_{\boldsymbol{\alpha}} \ \hat{f}(\boldsymbol{\alpha})$

    Evaluate true function for found $\boldsymbol{\alpha}_{\min}$: $g_{0,\min} = f(\boldsymbol{\alpha}_{\min})$

    Append samples with new sample:
      $\boldsymbol{A} \leftarrow \boldsymbol{\alpha}_{\min}$
      $\mathbf{g}_0 \leftarrow g_{0,\min}$

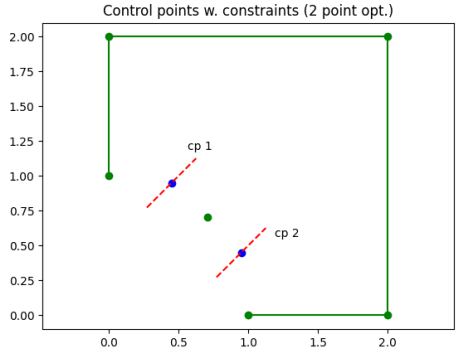
    Calculate residual: res $= \frac{g_{0,\min} - g_{0,\min}^{\text{old}}}{g_{0,\min}^{\text{old}}}$
  end while

4.1.3 Method

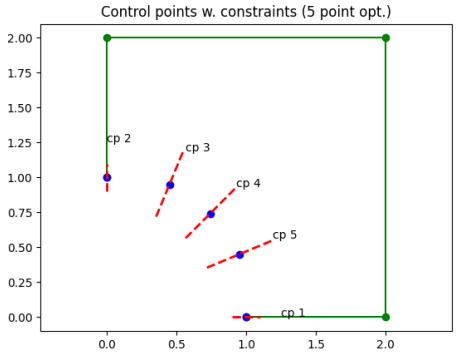
Implementation of the benchmarking test was done entirely in python with the use of CALFEM for python for mesh creation and FE-routines. Complementing this were self written DOE/sampling, surrogate model and optimisation scripts. The methodology for the benchmarking test began with the implementation of an FE-simulation using CALFEM, creating and meshing a geometry defined by control points, applying the loads and extracting the results. Secondly, two sampling algorithms were implemented, both LHS and HSS. A script producing a second degree full interaction polynomial surrogate model given sample points was written. The surrogate model was able to return a minima through an internal iteration using gradient descent starting on the minima of the sample points. These algorithms were then joined into a shape optimisation main algorithm as described in algorithm 3. Optimisations were performed for 2-, 5- and 7 control point geometries. For each optimisation, $A_{\max} = 4 - \frac{\pi}{4}$, which in theory would produce a hole with a radius of 1 length unit. The results can be seen in following sections.

(a) Box constraints for the problems

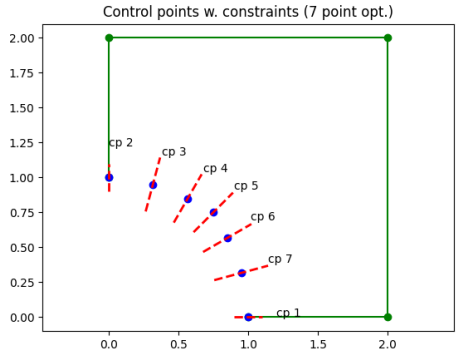
Opt. problem	Box constraints on $\boldsymbol{\alpha}$
2-point	$\pm[0.25,\ 0.25]$
5-point	$\pm[0.1,\ 0.1,\ 0.25,\ 0.25,\ 0.25]$
7-point	$\pm[0.1,\ 0.1,\ 0.2,\ 0.2,\ 0.2,\ 0.2,\ 0.2$



(b) Visualisation of the box constraints for 2-point optimisation



(c) Visualisation of the box constraints for 5-point optimisation



(d) Visualisation of the box constraints for 7-point optimisation

Figure 4.2: Design variable box constraints for the benchmarking optimisation problems.

Algorithm 3 Benchmarking shape optimisation algorithm

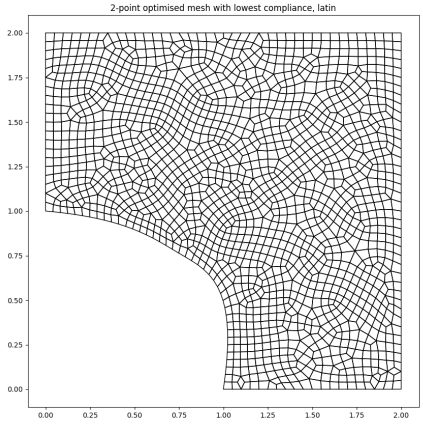
Initialisation
Define undeformed control points P_0
Define control point constraints b
Initialize design variables $\boldsymbol{\alpha}$
$n_{cp} \leftarrow$ number of control points
$n_{dv} \leftarrow$ number of design variables
Define max volume $V_{\text{opt}} \leftarrow 4 - \pi/4$
Set volume penalty factor μ

DOE
Define number of samples n_s
Choose sampling strategy *type* (LHS or Hammersley)
Generate samples:
$\mathbf{S} \leftarrow \text{DOE}(n_{cp}, n_s, b, type)$
$N \leftarrow$ number of samples
Initialize $g_0 \leftarrow [\,]$
for $sample = 1$ to N **do**
  $\alpha \leftarrow \mathbf{S}[sample]$
  $P_c \leftarrow \text{cp\_construct}(P_0, \alpha)$
  Run simulation and extract compliance, volume: $C, V \leftarrow \text{biaxialPull}(P_c, n_{\text{elm}}, force\_mag)$
  Calculate objective: $g_0[sample] \leftarrow C + \mu(V - V_{\text{opt}})^2$
end for

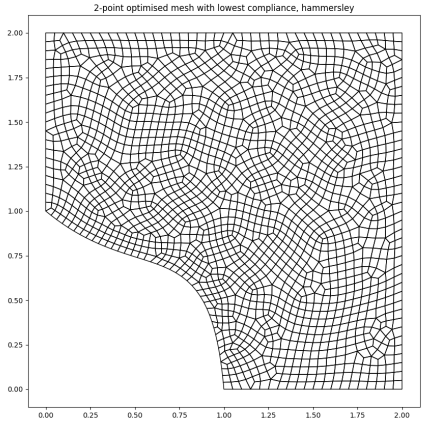
Optimisation loop
$g_{\min} \leftarrow \min(g_0)$
Set learning rate lr, tolerance ε
$k \leftarrow 0$
while $res > \varepsilon$ **do**
  Find minima on surrogate: $\boldsymbol{\alpha}_{\min}, g_{\min}, \hat{\boldsymbol{\beta}} \leftarrow \text{surrogate\_PRG}(\mathbf{S}, g_0, b, lr)$
  $cp \leftarrow \text{cp\_construct}(P_0, \alpha_{\min})$
  Run simulation and extract compliance, volume: $C, V \leftarrow \text{biaxialPull}(cp, n_{\text{elm}}, force)$
  $\mathbf{S} \leftarrow \mathbf{S} \cup \{\alpha_{\min}\}$
  Update objective: $g_{\text{new}} \leftarrow C + \mu(V - V_{\text{opt}})^2$
  $g_0 \leftarrow g_0 \cup \{g_{\text{new}}\}$
  Check convergence of minima: $res \leftarrow \left| \frac{g_{\text{new}} - g_{\min}}{g_{\min}} \right|$
  $g_{\min} \leftarrow g_{\text{new}}$
  $k \leftarrow k + 1$
  if $k > max\_itr$ **then**
    break
  end if
end while

4.1.4 Results

A wide range of optimisations with varying number of control points, sampling methods and sampling points were performed. The for each problem and sampling method, the optimised mesh with the lowest compliance can be seen in Figures 4.3 through 4.5 below. Data regarding optimisation iterations until convergence as a function of initial sampling points can be seen in Figure 4.6,4.7 and 4.8. As seen in Figure 4.3, 4.4 and 4.5, none of the meshes are circular. Looking at Figures 4.6, 4.7 and 4.8, LHS varies greatly in number of optimisation iterations for the same number of initial samples, whereas HSS is consistent, except for 30 IS points in the 5-point problem, and 30 and 75 IS points in the 7-point problem. Overall, HSS appears to need fewer, or roughly the same within the standard deviation, optimisation iterations for a given number of IS points. For tables containing data of all optimisations runs, see Appendix B.

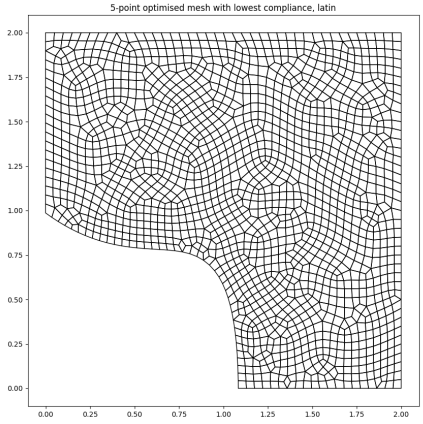


(a) Best performing LHS mesh, 20 IS points.

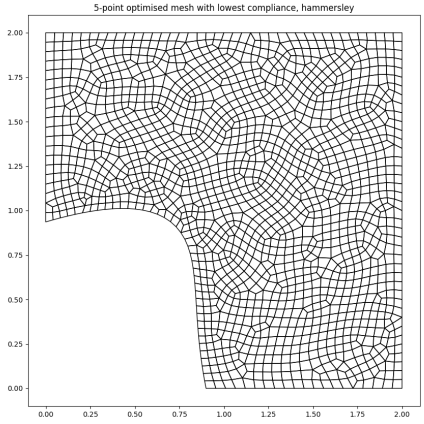


(b) Best performing HSS mesh, 30 IS points.

Figure 4.3: Meshes of what the optimiser found to have minimal compliance, 2-point problem.

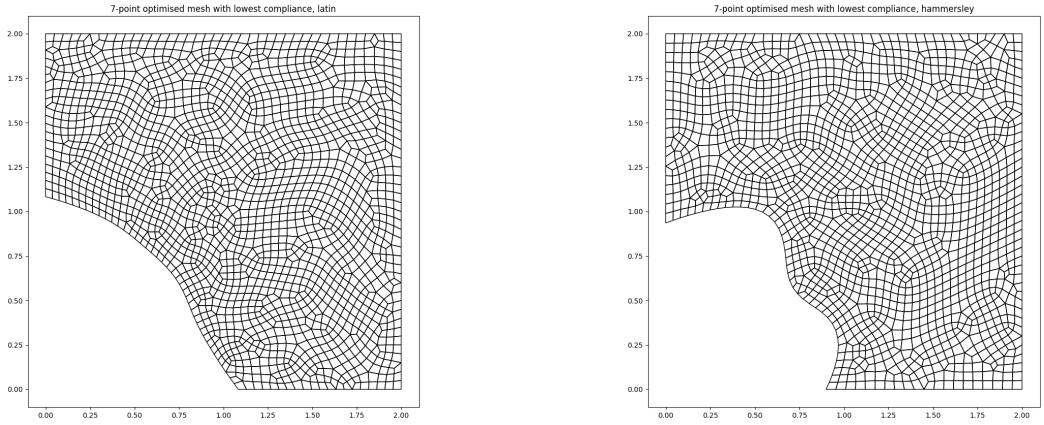


(a) Best performing LHS mesh, 75 IS points.



(b) Best performing HSS mesh, 75 IS points.

Figure 4.4: Meshes of what the optimiser found to have minimal compliance, 5-point problem.



(a) Best performing LHS mesh, 50 IS points.

(b) Best performing HSS mesh, 75 IS points.

Figure 4.5: Meshes of what the optimiser found to have minimal compliance, 7-point problem.

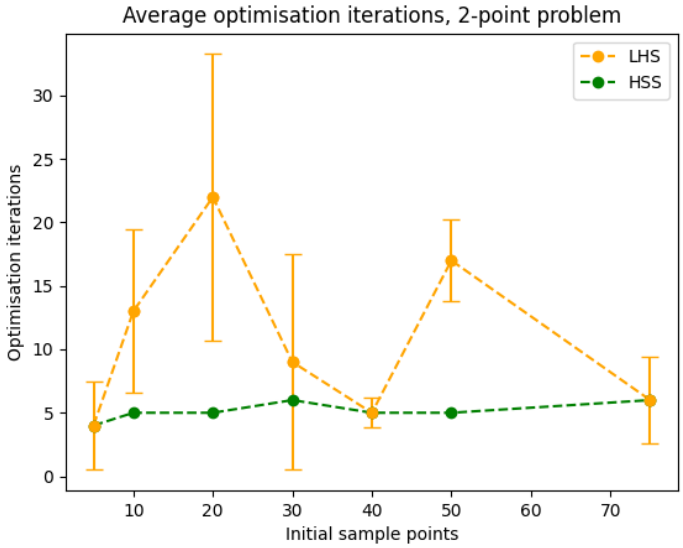


Figure 4.6: Average optimisation iterations against initial sample points for the two sampling methods, 2-point problem.

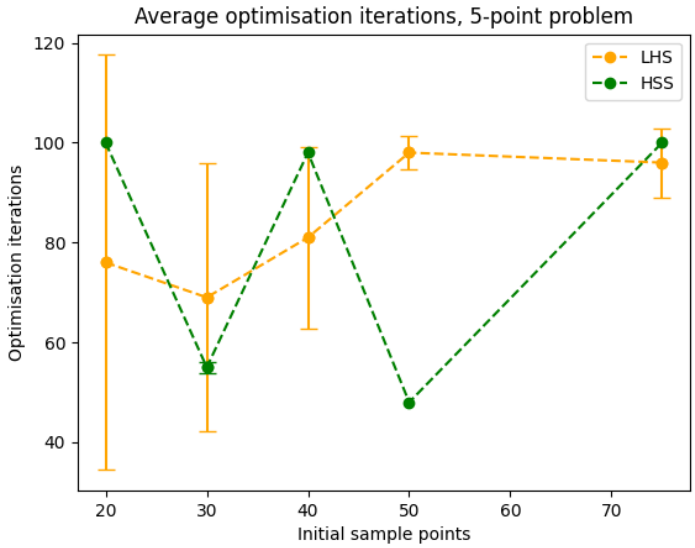


Figure 4.7: Average optimisation iterations against initial sample points for the two sampling methods, 5-point problem.

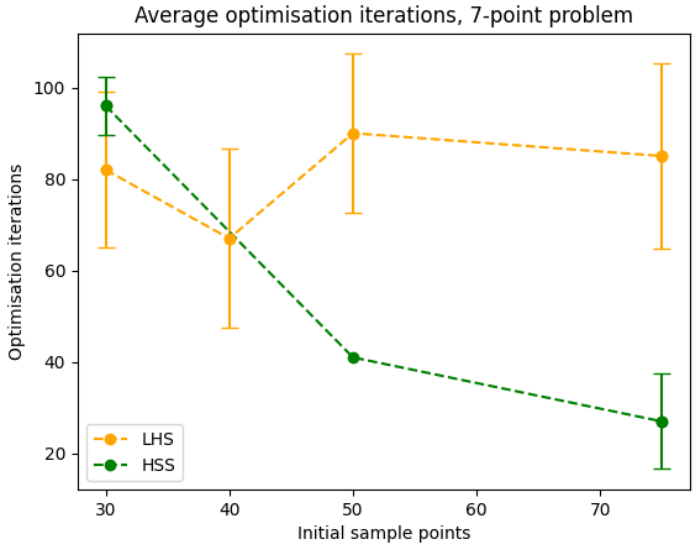
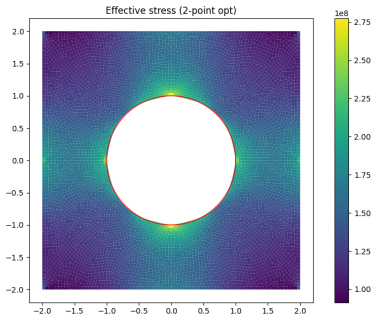


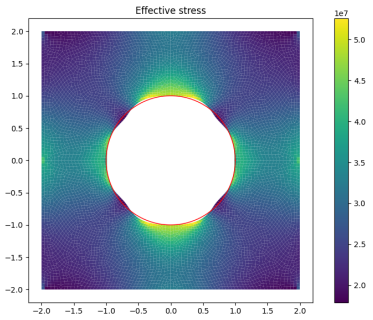
Figure 4.8: Average optimisation iterations against initial sample points for the two sampling methods, 7-point problem.

4.1.5 Discussion

As the figures with the meshes reveal, none of the optimisations resulted in a perfect circle with a radius of one. This came as a surprise after performing all optimisations in a single loop, as initial tests with the benchmarking worked fine, and results such as can be seen in Figure 4.9 were commonplace. It would appear that something went awry in these simulations. As the results seen above were produced at a later stage, when implementation of the Bayesian optimisation methodology was underway, it was no priority to perform any fault analysis.



(a) 2-point problem.



(b) 7-point problem.

Figure 4.9: Figures from initial testing phase of benchmarking with effective stress distribution of optimised mesh

The initial phase of benchmarking showed that optimisation with surrogate modelling was viable, even though the results presented here makes the statement questionable. Nevertheless, it provided experience with both surrogate modelling and different initial sampling methods. In fact, the independent implementation of both LHS and HSS were later repurposed for the Bayesian optimisation methodology.

One of the main lessons learned from the benchmarking was that surrogate modelling using regression was not ideal for use in a design space where the functional form approximation does not hold. Forcing a second order polynomial to data that does not follow that approximation creates a model which is both inaccurate and disregards truth. The inability to interpolate the model to the sampling points, which are seen as ground truth due to being FE-simulations, is a clear drawback. This could be solved by constraining the design space, ensuring a greater probability of a valid convex approximation. However, for problems with highly non-linear and discontinuous behaviour, such as contact problems with advanced constitutive models, this means that the initial design must be, approximately, the optimised design. As for the scope of this thesis, where an optimisation method should act on a black-box simulation and optimise to a desired response, this indicated that more advanced surrogate modelling based optimisation methods were needed.

4.2 Initial Testing in HyperWorks

The initial project scope was to perform the optimisation in commercial software used by Tetra Pak. The reason for this was to simplify the methodology, reduce the amount of scripting required, and provide a solution that is easy to hand over to the company. The main idea of the methodology was to set up a simulation in HyperMesh which included parameters that could be used as design variables. The model could then be imported into HyperStudy, in which the optimisation would be performed. The simulation setup and the optimisation methodology are described in further detail in the following sections.

4.2.1 Simulation Setup

The simulation for the optimisation problem was set up in HyperMesh 2026 with the Abaqus explicit solver profile. The geometry consisted of a press, paperboard and an anvil, as can be seen in Figure 4.10. All components were meshed using CPE4R four-node elements. Both the anvil and press were specified as rigid bodies, a valid assumption in relation to the paperboard. The paperboard was defined using a tuned simple hyperfoam material model belonging to Tetra Pak and will not be specified further. A symmetry condition was placed along the left boundary, with the leftmost nodes of the paperboard being constrained in DOF 1. As rigid bodies, both the anvil and press were constrained in all DOFs, except DOF 2 for the press, in which a concentrated force of 2 N was applied to a reference node. Frictionless general contact was applied to the contact pairs press-paperboard and paperboard-anvil, both with a penalty parameter of 200. A dynamic explicit force controlled load step was constructed, with a simulated time of 5 ms.

In order to introduce design variables to the simulation, the anvil had to be able to deform according to a set of control points. This proved to be a challenge with HyperStudy, and will be described further in the discussion section. In the end, design variables were able to be introduced using morph shapes, the effect of which can also be seen in Figure 4.10. By first creating an additional sketch of a b-spline with parameters for the control points, a line could be created in the geometry. By applying a displacement to each of the control points, the spline could be changed. Then, a morph of the mesh could be made to the deformed spline and recorded as a morph shape. This resulted in a set of morph shapes acting as design variables, each approximating the effect of each control point in the spline. The morph shapes could then be applied in weighted combinations to represent certain configurations of design variables.

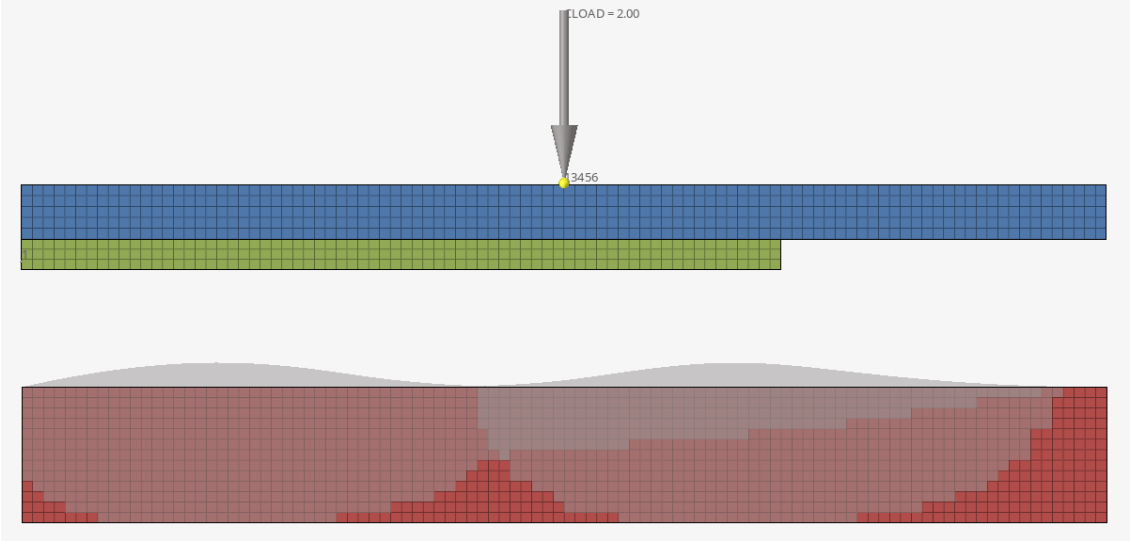


Figure 4.10: HyperMesh model for initial testing with elements affected by morph shapes 1 and 4 highlighted in grey.

4.2.2 Optimisation Setup

The concept for the optimisation methodology created for the initial testing is described below. The methodology included a number of steps distributed across different HyperWorks software, see Figure 4.11 for future reference. The initial step is creating the model simulation in HyperMesh. At the same time, a conscious choice or creation of design variables, that will later be used in the optimisation, must be made, e.g. using morph shapes. Subsequently, a new HyperStudy study is set up, where the HyperMesh model is imported. Here, the design variables are selected, as well as what output responses to generate. These output responses can be created to evaluate the objective function.

Then, a DOE is performed and an analysis using Abaqus launched via HyperStudy is performed for each of the sample points. In HyperStudy, many choices exist for the DOE methods, such as the familiar LHS, HSS and SSS, as well as other methods, such as factorial and fractional factorial methods. After having performed a DOE, a surrogate model can be constructed using Fit Model. Here, choices of least square, moving least square, random forest and HyperKriging exists, to name a few.

After the surrogate model is constructed, an optimisation can be performed. An objective can be defined using a single or a combination of output responses, as well as other data sources. Then, a method for optimising on the surrogate model can be chosen. Here, many gradient based choices can be made, both local and global. After a method has been chosen, a maximum number of optimisation iterations can be chosen, and then the optimisation can begin. After completion, HyperStudy marks the iteration that has the best value with respect to the objective function, highlighting the design variable values. These values can then be input into the HyperMesh model, and a validation of the result can be made.

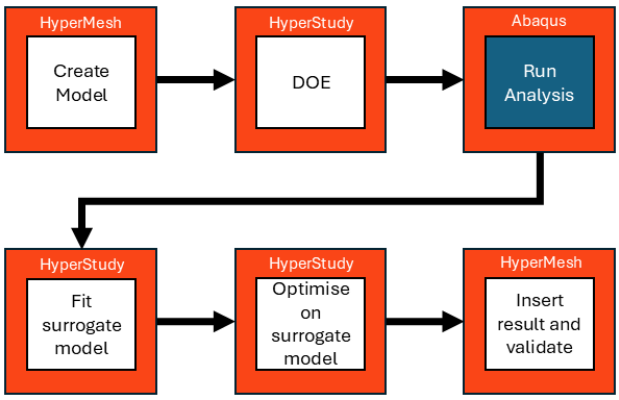


Figure 4.11: Flowchart of HyperStudy optimisation workflow

4.2.3 Discussion

After spending several weeks working with HyperStudy, using different model fits, optimisation methods, reducing the number of design variables et cetera, it was decided that HyperStudy was not the correct choice of commercial software to move forward. The optimisation workflow for shape optimisation that was attempted in HyperStudy simply could not produce any results of significant value, as well as being too limiting in freedom of customisation. Short discussions regarding each of the problems that arose can be read below.

To begin, introducing design variables related to a B-spline became a challenge. Initially, in HyperMesh, the geometry of the anvil was parametrised with a B-spline created as a sketch. The parameters could then be varied, morphing the geometry. However, the mesh did not follow the same morph. The plan was then to introduce a script in HyperStudy that would mesh the morphed geometry and allow the analysis to be run. However, when importing the HyperMesh model to HyperStudy, these geometry parameters could not be chosen as design variables and caused the program to crash. After trial and error, a solution was found using morph shapes, where a morph could be applied to the mesh directly. An underlying B-spline was created and morphing the spline using one control point at a time, then morphing the mesh to the spline and saving it as a shape, the approximate effect of each control point could be recorded. These would then be able to be applied in combination, resulting in different designs. Additionally, morph shapes were valid as design variables when imported to HyperStudy, showing promise for the shape parametrisation.

Extracting the results and evaluating the objective function proved to be another challenge. Two choices existed, incorporating a python script into HyperStudy, which allowed the result file to be read and relevant results extracted, or using a built-in reader to read the result file. The latter was chosen to reduce time and effort spent. Initially, the built-in reader performed well, but problems arose after a while when HyperStudy seemed to encounter a software glitch and suddenly seemed to forget that the reader existed. These results were therefore unsuccessfully extracted, and the only way to make HyperStudy recognise the built-in reader was to restart the software. As the optimisation methodology should require as little manual labour as possible, this proved frustrating.

As proved by the benchmarking example, using simple surrogate models, such as PRG, proved insufficient for reliable optimisation. After discussions with Dr. Johan Lindström at the division of mathematical statistics at LTH, it was decided that Bayesian optimisation was an interesting methodology to apply instead. The problem with Bayesian optimisation was that it could not easily be integrated in to HyperStudy. The initial sampling was possible, and HyperStudy had a model fit named HyperKriging, but it was unclear if this was an actual Gaussian process or a

simplified version. Since the model fit was unclear, this would require scripting. Furthermore, adding additional points after the initial sampling with the help of acquisition functions would require scripting, as the choice for adding new points went through an additional DOE, extending the initial sampling. Lastly, the built-in optimisation methods for the surrogate model behaved in a manner that was unfamiliar. The optimiser evaluated a predetermined number of points (referred to as iterations) and then identified the best-performing one. For us, being accustomed to gradient-based optimisation with convergence criteria, this approach was unfamiliar. In summary, the optimisation methodology that we wanted to implement would require scripting to a large extent, which was made harder by the fact that the scripting would have to be integrated properly with the software.

As a results of these problems, it was decided to not move forward in HyperStudy and not risk spending any more time on an implementation whose ability for optimisation was unclear. As customising HyperStudy to fit the needs of the methodology to a large extent would require scripting, the choice was made to switch to a fully scripted methodology in python. Writing an independent implementation was thought to reduce the potential problems with integrating software and script, as commercial software only would be needed to start and analyse a simulation. The methodology developed is described in chapter 5.

It is important to note that the decision to discard the use of HyperStudy in this thesis in favour of custom scripting does not invalidate or diminish its use case. HyperStudy remains a software with a low barrier for use. It allows the input of a simulation setup and a simple way to perform a screening and optimisation with varying complexity.

Chapter 5

Bayesian Shape Optimisation Methodology

5.1 Overview

As previously stated, the Kriging model is simply an input-output function, and the data can be generated from a black-box function. Since there are no physical laws present, there is freedom in what to include in this black-box function. To reflect this modularity, a Bayesian optimisation methodology has been implemented in Python with the ability to act on an arbitrary black-box function, see the flowchart in figure 5.1. In python, the design variables and their bounds are set, i.e. the input to the Kriging model, and sent into the black-box. The black-box then generates an output value which the Kriging model is fit to. This function can be set to do whatever the user wants.

In the case of the application in this thesis, the black-box is represented by a function that turns design variables into a g_0 using the pressure distribution from a simulation. Firstly, the function creates an anvil using the design variables as weights for the B-splines. This anvil is then turned into an .inp-file, and combined with the Abaqus-model created in HyperMesh. In python, the initial sampling method, number of sample points, acquisition function, and number of acquisitions are set. These conditions determine how the initial .inp-files are created. The finished inp-files contain the anvil, created in python, together with the paperboard and the sonotrode, modelled in HyperMesh. These inp-files are then run using the Abaqus Explicit solver and their pressure is extracted with a separate python script. The pressure is then recomputed into g_0 and fed into the Kriging model. If no acquisition phase has been set, the model is optimised upon and a result is found.

Otherwise, the acquisition phase begins. In this phase, the acquisition function chooses a singular point, or a number of points, to be evaluated and fed into the model. The model is then refit and more points are elected until the model has "converged". In the current model, convergence is determined by a fixed sampling budget. Once the acquisition phase is over, the model is optimised upon and an .inp file containing the optimal anvil is generated. The performance of this anvil is then validated manually running a separate python script running the simulation. Except for this validation step, all steps after parameter definitions and optimisation preferences are performed automatically.

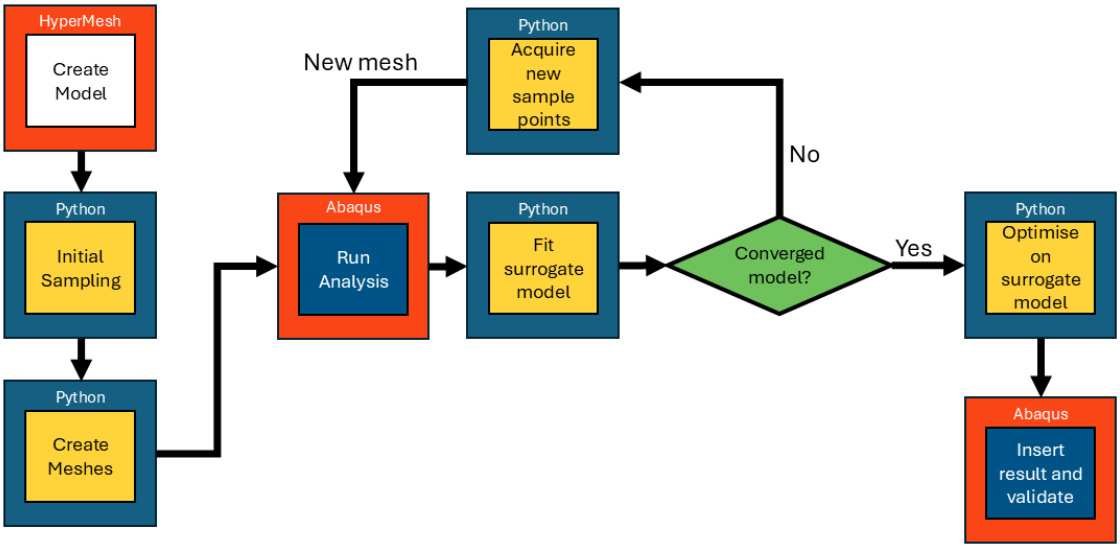


Figure 5.1: Flowchart of Bayesian shape optimisation methodology

5.2 Implementation

Python was used for implementing the methodology. To the largest reasonable extent, this was achieved with self-written code. However, for the implementation of the Kriging modelling - with fit and hyperparameter optimisation - and acquisition functions, the python library BoTorch was used. For the implementation of a GP in BoTorch x- and y-data should be handled differently. For the GP to function the best all x-values shall be *normalised* to $[0,1]^d$, meaning all values lie between zero and one, and all y-values should be *standardised* to [0,1], meaning the mean is zero and standard deviation one. When applied to the anvil, the data is neither normalised nor standardised. Therefore the terms physical and virtual space was used to differentiate between them. While the x-data has to be transformed to the virtual space before input, the y-data is transformed internally. The model therefore also returns x-data in the virtual space and y-data in the physical space. To clarify: the x-data has to be *normalised* before inputting them to the GP, while the y-data is *standardised* within.

Additionally, BoTorch does not include any acquisition functions that minimise the function. therefore the objective functions sign was reversed to allow the use of maximising acquisition functions. For the optimisation, the maximum of the reverse sign values was found, which when reversed back again are presented as the minima of the function. This is a quirk of BoTorch and nothing that affects the minimisation. However, it can generate unphysical predicted g_0 values which are negative, but that is because the optimiser found a positive value as the maximum. These unphysical values may be present in the model, but in the validation, the true g_0 of the optimised solution is found.

Fitting the Kriging model can be problematic. At times, during testing, the model was not able find an adequate fit. To solve this issue, an artificial internal noise was added to the y-values in the virtual space. Currently, the standard is 10^{-4} but has to be increased when the model fails. In the case of failure, the noise was increased to 10^{-2}, to increase the chance of success. The level of noise has not been optimised. Different levels were tried, and once levels were found that *worked*, they were chosen as standard.

Chapter 6

Optimisation of Ultrasonic Sealing Anvil

As a test and proof of concept of the methodology, it is applied to shape optimisation of the anvil in the ultrasonic sealing process. The simulation is simplified and in 2D, as it is a reasonable starting point. In the first section, the simulation and the optimisation problem are described. In the following section, the method and different tests performed are detailed. Thereafter, the results are presented. The last sections acts as discussion for both the results as well as the entire thesis and outlines suggestions for future work.

6.1 Problem and Simulation Description

Simulation

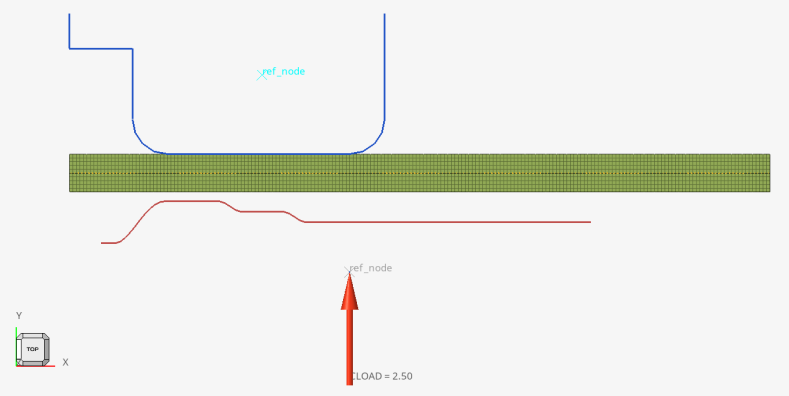


Figure 6.1: Screenshot of simulation setup. Contact region marked with yellow dots, where the pressure distribution is analysed.

A screenshot of the simulation can be seen in Figure 6.1. The simulation is comprised of four main components. They are the sonotrode in blue, the anvil in red, and the upper and lower layer of packaging material in green. The upper and lower layers of packaging material appear as a single component, but they are not connected and are simply in contact at the start of the simulation. The sonotrode and anvil are both treated as rigid bodies due to the relative stiffness of titanium compared to the hyperfoam. As only contact phenomena on the boundary of these components are interesting for the simulation, they are modelled using a line mesh. The sonotrode has 42 elements, of which the lowest 21 are included in the contact calculations, which are specified later.

The anvil is defined in a separate script with 100 nodes, resulting in 99 elements, of which all are included in the contact calculations. Each layer of packaging material is modelled using 1000 square S4R elements each, with 5 elements in height and 200 in width. The material model for the packaging material layers is the same hyperfoam, tuned by Tetra Pak, as used in initial testing.

There are several boundary conditions applied to the components. The sonotrode is constrained in all 6 DOFs and remains stationary throughout the simulation. The anvil is constrained in all DOFs except DOF 2, allowing it to move in the y-direction. Due to the symmetry present in the real world application, with two seals made simultaneously, there is x-symmetry in the leftmost boundary of the packaging material. These nodes are thus constrained in DOF 1, as well as DOF 5 and 6. Further, to prevent any errors with the 2D analysis, all nodes in the packaging material is constrained in DOF 3,4 and 5 as well.

Abaqus general contact is applied in all contact interaction pairs. These contact pairs are defined using sets of element surfaces. The packaging material layers each have two contact sets. They include all element surfaces on the top boundary as well as the bottom boundary. Contact pairs are made between the anvil and lower packaging material bottom surface, lower packaging material upper surface and upper packaging material bottom surface, and upper packaging material upper surface and the sonotrode. In the general contact, the scale factor, acting as a penalty parameter, is set to 200.

The load step consists of force controlled loading applied to the anvil in the positive y-direction. The explicit solver profile in Abaqus is used, as to align with simulations at Tetra Pak. A ramp function is used on the force, beginning at 0 N and increasing linearly over 10 ms to 2.5 N, which is the end time of the simulation. At the final time step, the contact pressure between the two layers of packaging material is extracted and used to compute g_0. A screenshot of such a final time step can be seen in Figure 6.2, with all contact pressures of the packaging material boundaries displayed.

To reduce computational effort, while still providing a proof of concept of the methodology, the simulation is significantly simplified compared to the real process. This simulation effectively only models the application of the preload on the packaging material and investigating the resulting pressure distribution. A more refined model would include the effects of the sonotrode vibrations and have a more realistic representation of the packaging material with several layers of paperboard and polymer. Additionally, as it is just a proof och concept and not a design study, the model is also 2D, whereas the real anvil design would require 3D optimisation. However, since the methodology treats the simulation as a black-box function, it would be able to handle more advanced simulations as well.

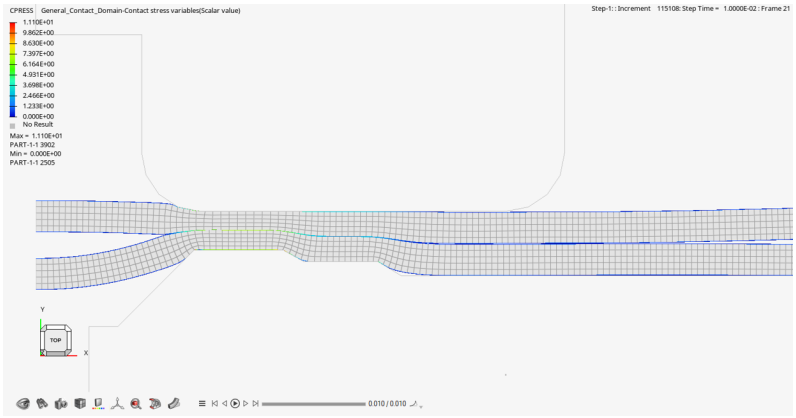


Figure 6.2: Screenshot of final time step using an arbitrary anvil. Contact pressures of packaging material are displayed.

Optimisation Problem

The shape optimisation problem for the sealing anvil is as follows:

$$\text{SO}: \begin{cases} \min\limits_{\mathbf{x}} \;\; g_0(\mathbf{x}) \;\; = \sum\limits_{\text{node } i} (P_i(\mathbf{x}) - P_{\text{ref},i})^2 \\ \\ \text{such that} \;\; x_i \in [-x_i^{\max},\; x_i^{\max}] \end{cases} \tag{6.1}$$

The objective function is the squared norm of the nodal contact pressures with the nodal contact pressures of the reference pressure distribution. The spline control points, acting as design variables, are constrained by box constraints. The objective function should be minimised, but in practice, the negative objective function is maximised. The reason for the flip is due to BoTorch, in which the acquisition functions are more adapted for maximising. The principle is still the same, as values close to zero are seen as desirable. For the 1-,2- and 5-variable tests, a reference pressure distribution is generated using a reference anvil with chosen design variable values. For the full anvil optimisation, an arbitrary pressure distribution was created, meaning the true anvil is unknown, thus making the test more realistic.

6.2 Method

This section outlines what optimisations were performed and how they were set up. Throughout the early testing these models proved to be sensitive. Fitting the model could be difficult when the function values varied too much in close proximity. The following simulation were set up to test how the method code performs under certain conditions, finishing with a full anvil optimisation.

1D Screening

In order to understand the problem and objective function further, a screening of five distinct design variables was performed. The chosen basis functions are located in five interesting regions of the anvil, with two on the flat peaks and one each for the slopes of the peaks. The selected design variables and their respective constraints for the screening can be seen in figure 6.3. To perform the screening, each of the design variables were simulated for 20 equidistant points in the space $\alpha_i \in [-0.2,\; 0.2]$.

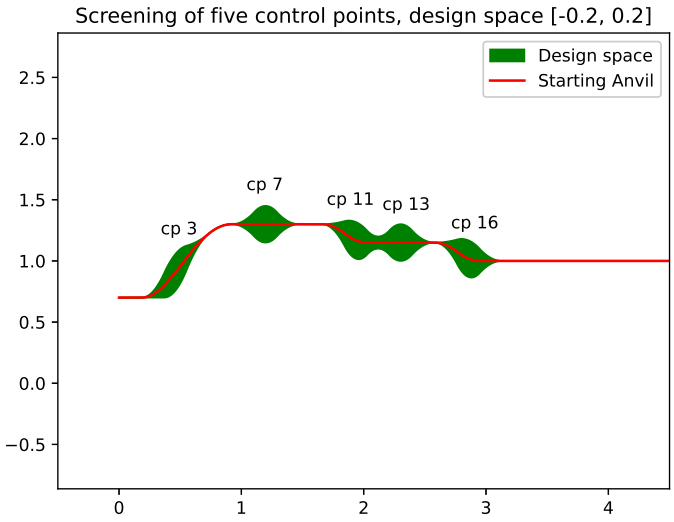


Figure 6.3: Spline representation of anvil with design variable and constraints selected for screening highlighted.

One Variable Optimisation

This initial test aimed at proving if the methodology is capable of finding large shape changes in a single dimension. To test this we chose the most influential basis from the screening as design variable. We then chose $\alpha_{opt} = 0.1$ as the deviation, as there was agreement it is a large shape change. So as to allow the model to make errors we set the design space to $\pm 2\alpha_{opt}$. To test the efficacy of different sampling budgets we chose qEI as the acquisition function and set number of points per batch to 5, then three different runs were performed, with 1,2, or 3 acquisition batches. For these runs an initial 10 sampling points with Hammersley sampling was used.

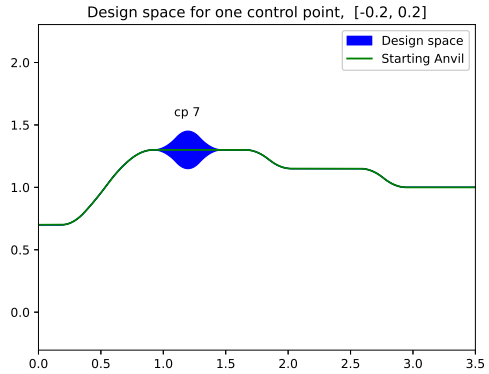


Figure 6.4: The design space for one variable optimisation with deviations in cp7.

Two Variable Optimisation

The purpose of these tests were to see the methodology functions in a low-dimensional space. To increase the chance of success we lower the optimal shape change to $\boldsymbol{\alpha} = [0.05, 0.03]$. The design space was set to ± 0.1. We also tested the efficacy of qEI and UCB by running qEI and UCB with two different values of $\beta, \beta = 2, 5$.

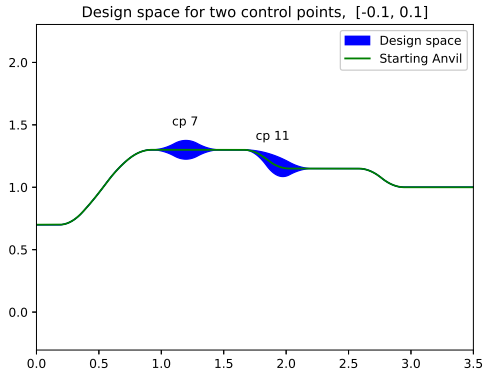


Figure 6.5: The design space for two variable optimisation with deviations in cp7 and cp11.

Five Variable Optimisation

These runs had multiple purposes, we wanted to see: if the methodology could optimise in a multidimensional space, how it handles bases with insignificant influence on g_0, how it handles overlapping bases, and lastly, how to different initial sampling methods perform against each other. To test these we designed nine different runs, three for each ISM. qEI was chosen as acquisition function with an additional 25 points being acquired. The reference was chosen as: $\boldsymbol{\alpha} = [0.05, 0.02, 0.03, -0.05, 0]$, with the designspace being ± 0.1.

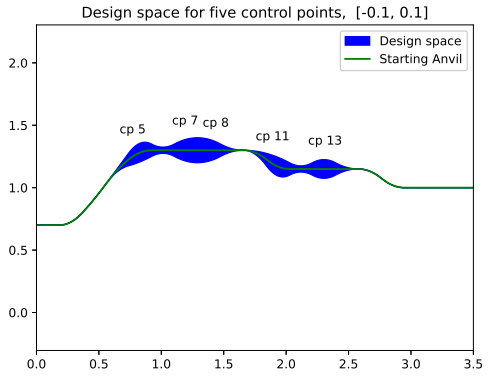


Figure 6.6: The design space for five variable optimisation with deviations in cp5, cp7, cp 8, cp11, and cp13.

Full Anvil Optimisation

To give the optimisation methodology a test in a less arranged setting, a pressure distribution was constructed using analytic functions. This meant that the anvil resulting in the reference pressure was truly unknown. All basis functions of the anvil between four and fifteen were included in the optimisation, resulting in 12 design variables. As the reference pressure was not constructed using a reference anvil, this was the first test for the methodology where success was not guaranteed to be possible. The pressure distribution can be seen compared to the starting distribution in figure 6.7, and the design space can be seen in figure 6.8.

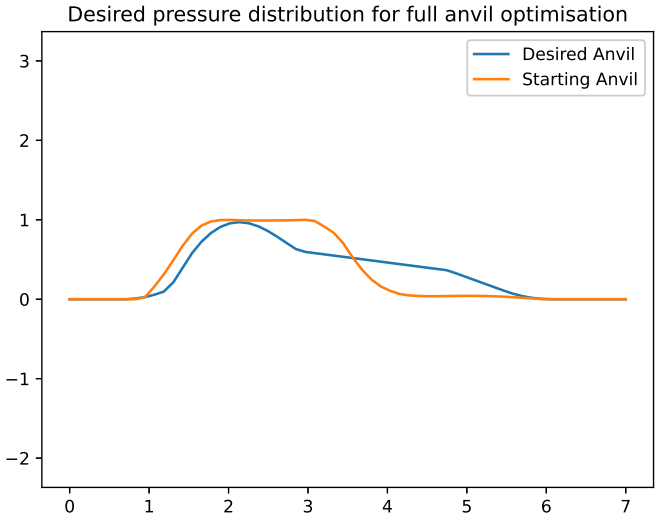


Figure 6.7: The desired pressure distribution and the pressure distribution of the starting anvil.

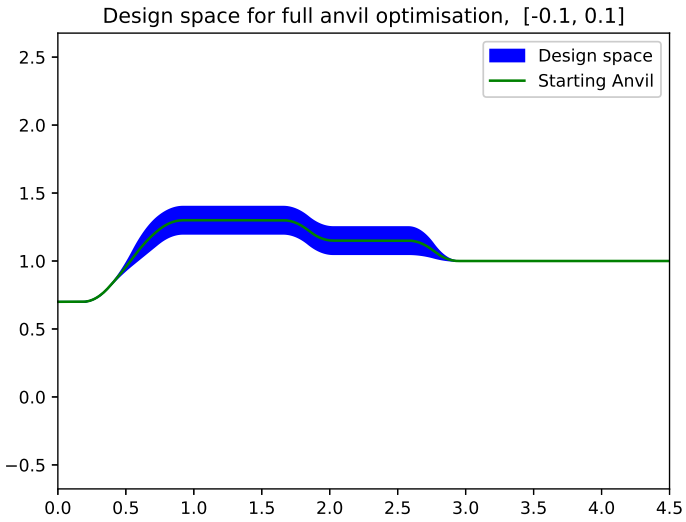


Figure 6.8: The design space for the full anvil optimisation.

6.3 Results

The results of the experiments can be seen summarised in table 6.1. For a more thorough presentation of the screening and the individual results of the experiments, such as resulting pressure distributions and anvil geometries, see the subsequent sections. For the pressure distribution and geometry figures, the whole dataset is not presented. Rather, only the region around nonzero pressure and the interesting region of the geometry is presented to aid the reader, as the full dataset would make differences challenging to see.

To evaluate how the optimisation of the posterior mean acts on the finished model, all sampled g_0 were compared to the best predicted g_0 in table 6.2. The posterior mean found values better than any sampled point in nine cases, whereas six cases had a lower g_0 in the acquisition points.

Table 6.1: The results of the optimisation experiments with known references.

Experiment run	**Design variables α**	**Predicted g_0**	**True g_0**
1D reference	[0.1]	-	-
HSS 10, qEI 5	[0.1398]	0.07359	0.1207
HSS 10, qEI 10	[0.1066]	0.06495	0.003877
HSS 10, qEI 20	[0.1267]	0.03494	0.02609
2D reference	[0.05, 0.03]	-	-
HSS 20, qEI 20	[0.05185, 0.02929]	-0.002544	0.0004678
HSS 20, UCB 20, β 2	[0.04530, 0.02679]	0.002992	0.003049
HSS 20, UCB 20, β 5	[0.04640, 0.02677]	-0.005383	0.002068
5D reference	[0.05, 0.02, 0.03, -0.05, 0]	-	-
HSS 25, qEI 25	[0.04415, 0.01134, 0.02427, -0.08090, -0.1000]	0.006343	0.01321
HSS 50, qEI 25	[0.04917, 0.01934, 0.02523, -0.04535, -0.04002]	0.0004896	0.008042
HSS 100, qEI 25	[0.05359, 0.01646, 0.03363, -0.02204, -0.05689]	0.005781	0.007500
SSS 25, qEI 25	[0.04056, 0.01567, 0.02756, -0.07859, -0.02811]	0.009844	0.01706
SSS 50, qEI 25	[0.02873, 0.01960, 0.02249, 0.05556, -0.07872]	-0.1205	0.02110
SSS 100, qEI 25	[0.03885, 0.01833, 0.02717, 0.01258, -0.05786]	0.008064	0.01264
LHS 25, qEI 25	[0.04449, 0.02488, 0.02684, -0.03138, -0.1000]	-0.04151	0.009924
LHS 50, qEI 25	[0.04036, 0.01361, 0.01885, 0.01887, -0.07739]	-0.004775	0.02745
LHS 100, qEI 25	[0.04222, 0.01362, 0.02465, -0.08832, -0.01200]	0.005492	0.006401

Table 6.2: The predicted g0 compared to the best sampled g0

Run	Best predicted g_0	Best sampled g_0
HSS 10, qEI 5	0.07359	**0.01576**
HSS 10, qEI 10	0.05495	**0.004353**
HSS 10, qEI 20	0.03494	**0.0005665**
HSS 20, qEI 20	**-0.002544**	0.0004227
HSS 20, UCB 20, β 2	0.002992	**0.002400**
HSS 20, UCB 20, β 5	**-0.005385**	0.003597
HSS 25, qEI 25	**0.006343**	0.009514
HSS 50, qEI 25	**0.0004896**	0.003872
HSS100, qEI 25	**0.005781**	0.006245
SSS 25, qEI 25	0.009844	**0.006014**
SSS 50, qEI 25	**-0.1205**	0.1688
SSS 100, qEI 25	**0.008064**	0.01059
LHS 25, qEI 25	**-0.4151**	0.008180
LHS 50, qEI 25	**-0.004775**	0.008072
LHS 100, qEI 25	0.005492	**0.005378**

6.3.1 1D Screening

The results of the 1D screening can be seen in figure 6.9 and 6.10. Figure 6.9 shows the raw data from simulations, where the objective function value is calculated using the pressure distribution of the unperturbed anvil, i.e. the anvil with zero design variable values, as reference. From the figure, it is clear that the seventh basis function of the spline representation has the most effect on the objective function. Basis three and sixteen have virtually no effect on the objective function over the screened design space, as the objective function is approximately zero except towards the upper end of the screening space for basis sixteen. Basis eleven and thirteen have results resembling each other, with the effect on the objective function mainly coming from the positive part of the screening space.

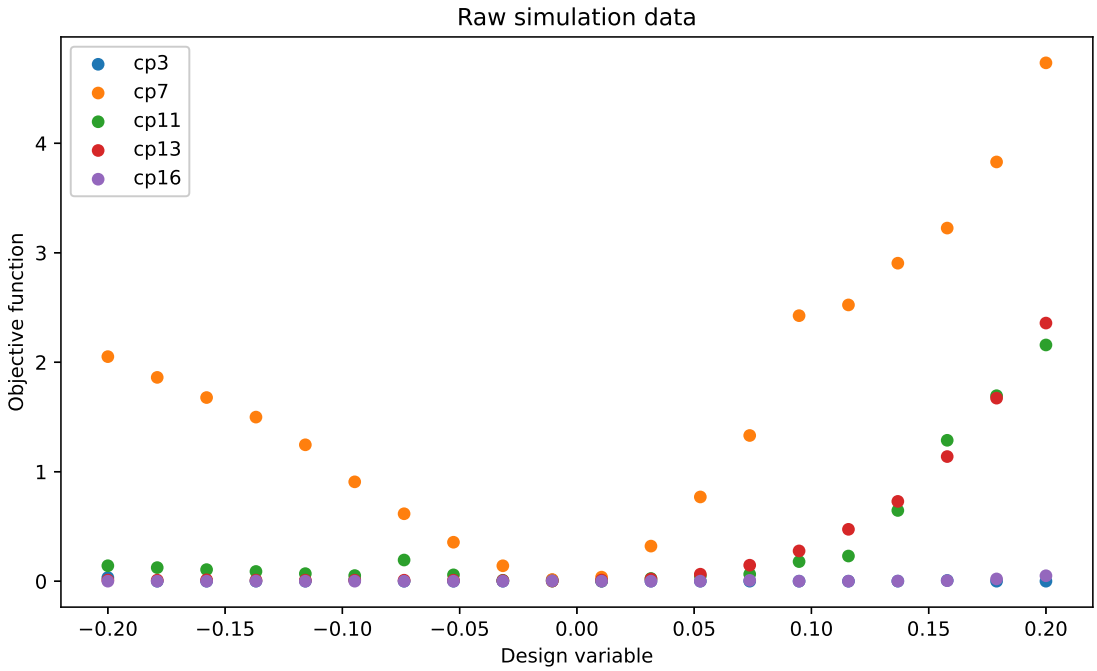


Figure 6.9: Raw simulation data of objective function value over screening design space for the five spline control points.

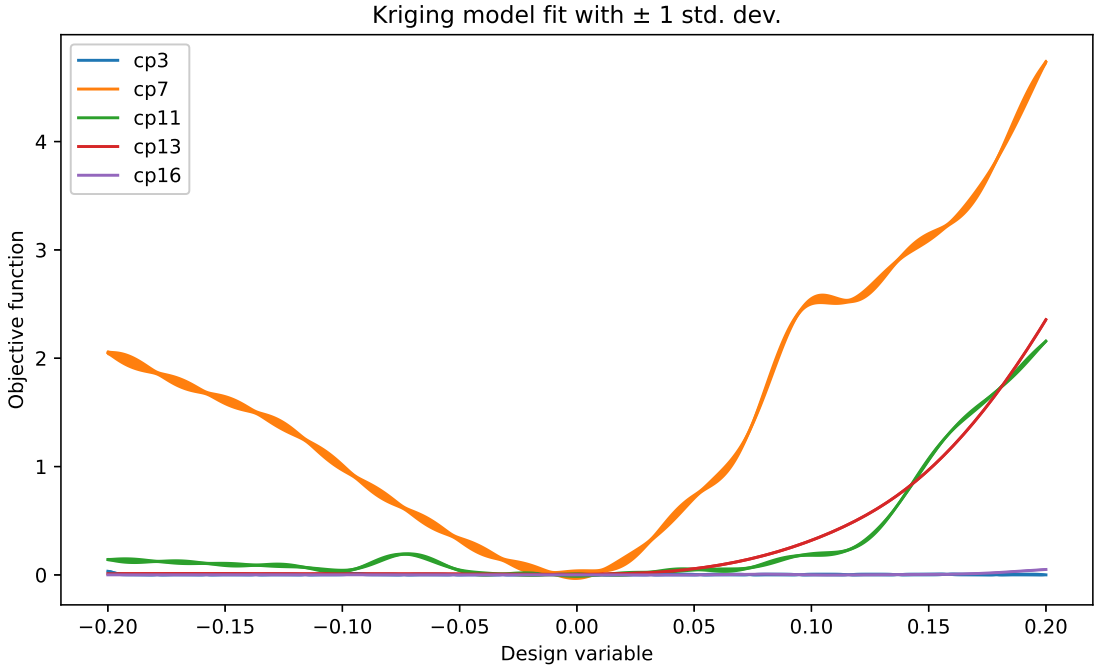


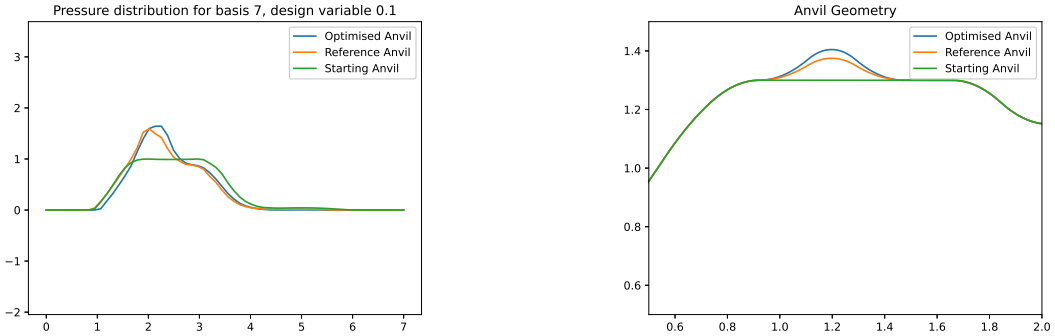
Figure 6.10: Model fit of raw data of objective function over screening design space for the five spline control points. Shaded region represents $\pm\sigma$.

6.3.2 One Design Variable

For these results, all runs were performed using 10 point HSS, varying only cp7 in the range $[-0.2, 0.2]$, with the reference being deviated by $[0.1]$.

HSS 10, qEI 5

The optimised weight was $\alpha = [0.1398]$ with a predicted $g_0 = 0.0736$ and a simulated $g_0 = 0.1207$ The resulting anvil and pressure curve can be seen in figure 6.11.



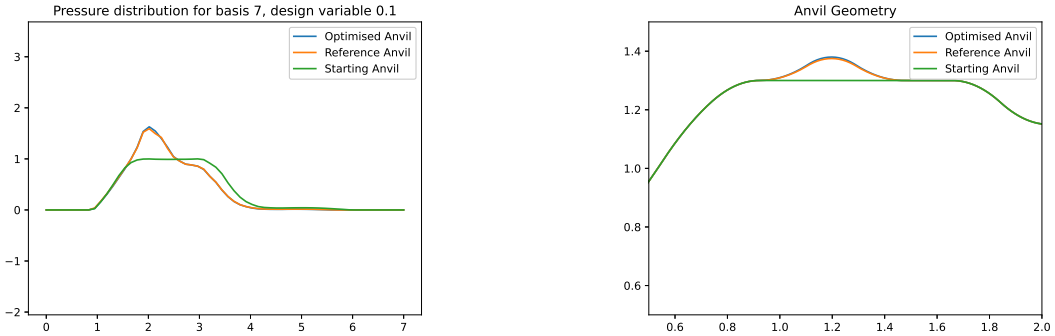
(a) The optimised pressure curve

(b) The optimised anvil

Figure 6.11: The results from running a 10 point HSS with an acquisition budget of 5 points

HSS 10, qEI 10

The optimised weight was $\alpha = [0.1066]$ with a predicted $g_0 = 0.0650$ and a simulated $g_0 = 0.0039$. The resulting anvil and pressure curve can be seen in figure 6.12.



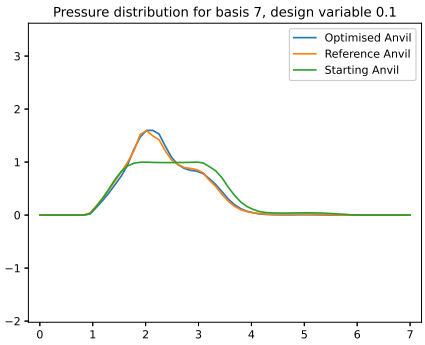
(a) The optimised pressure curve

(b) The optimised anvil

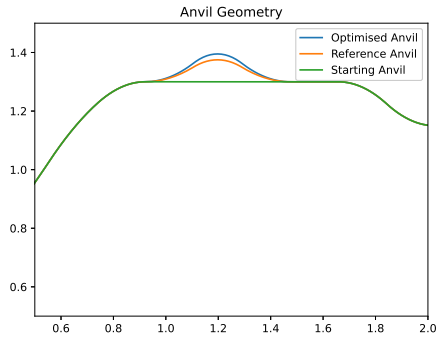
Figure 6.12: The results from running a 10 point HSS with an acquisition budget of 10 points

HSS 10, qEI 20

The optimised weight was $\alpha = [0.1267]$ with a predicted $g_0 = 0.0350$ and a simulated $g_0 = 0.0261$. The resulting anvil and pressure curve can be seen in figure 6.13.



(a) The optimised pressure curve



(b) The optimised anvil

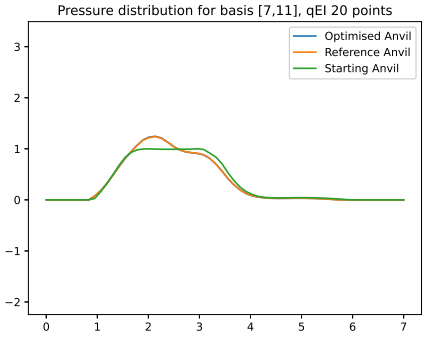
Figure 6.13: The results from running a 10 point HSS with an acquisition budget of 20 points

6.3.3 Two Design Variables

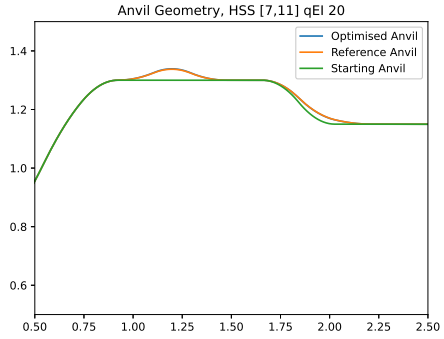
The following runs were all performed using 20 point HSS, varying cp7 and cp11 in the ranges [-0.1,0.1], with the optimal weights being [0.05,0.03].

HSS 20, qEI 20

Running 20 qEI batch acquisition points with HSS 20 point initial sampling for the 2D optimisation, the design variables became $\boldsymbol{\alpha} = [0.05185, 0.02929]$ with a predicted $g_0 = -2.544 \cdot 10^{-3}$ and simulated $g_0 = 4.678 \cdot 10^{-4}$.



(a) The optimised pressure curve

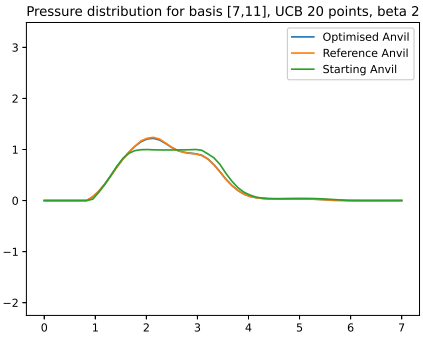


(b) The optimised anvil

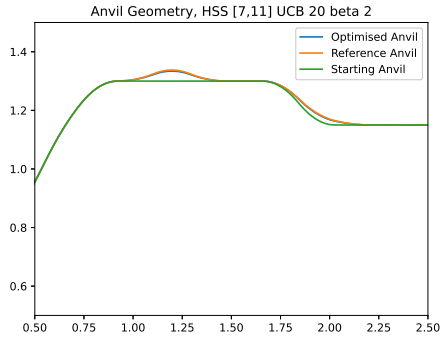
Figure 6.14: The results from running a 20 point HSS with an acquisition budget of 20 qEI points.

HSS 20, UCB 20, β 2

Running 20 UCB acquisition points with a $\beta = 2$ and HSS 20 point initial sampling for the 2D optimisation, the design variables became $\boldsymbol{\alpha} = [0.04530, 0.02679]$ with a predicted $g_0 = 2.992 \cdot 10^{-3}$ and a simulated $g_0 = 3.049 \cdot 10^{-3}$.



(a) The optimised pressure curve

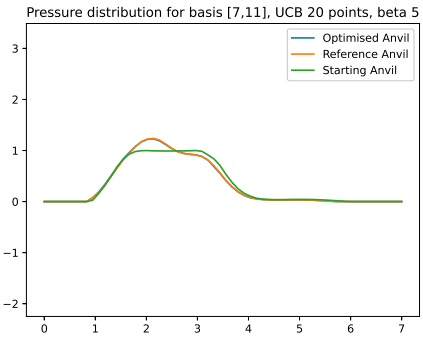


(b) The optimised anvil

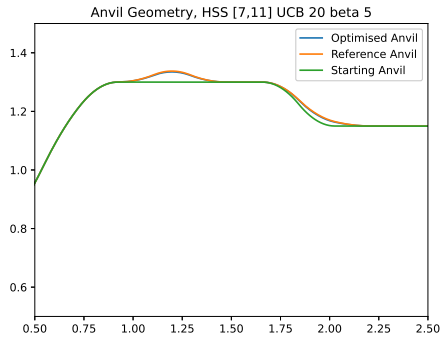
Figure 6.15: The results from running a 20 point HSS with an acquisition budget of 20 UCB points with $\beta = 2$.

HSS 20, UCB 20, β 5

Running 20 UCB acquisition points with a $\beta = 5$ and HSS 20 point initial sampling for the 2D optimisation, the design variables became $\boldsymbol{\alpha} = [0.0464, 0.02677]$ with a predicted $g_0 = -5.383 \cdot 10^{-3}$ and a simulated $g_0 = 2.068 \cdot 10^{-3}$.



(a) The optimised pressure curve



(b) The optimised anvil

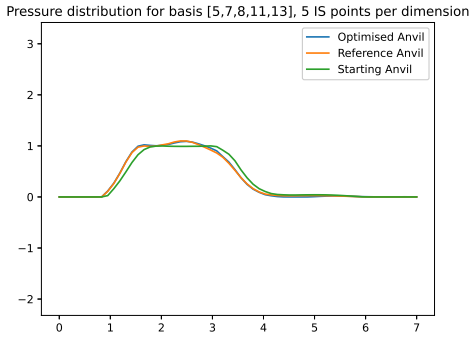
Figure 6.16: The results from running a 20 point HSS with an acquisition budget of 25 qEI points.

6.3.4 Five Design Variables

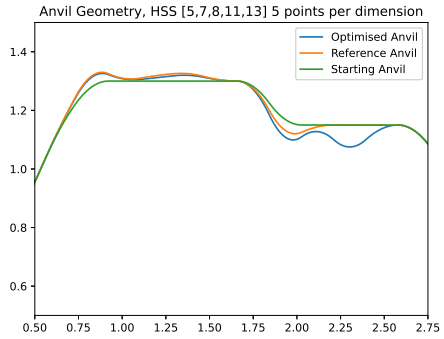
For the following runs the optimal design variables were: [0.05,0.02,0.03,-0.05, 0]. The runs were performed using HSS, SSS, and LHS as initial sampling methods, with a varying number of sample points. All aquisitions were performed with qEI and 25 acquisition points.

HSS 25, qEI 25

For this run the optimised design variables were: [0.04415, 0.01134, 0.02427, -0.08090, -0.1000] with a predicted $g_0 = 0.006343$ and a simulated $g_0 = 0.01321$



(a) The optimised pressure curve

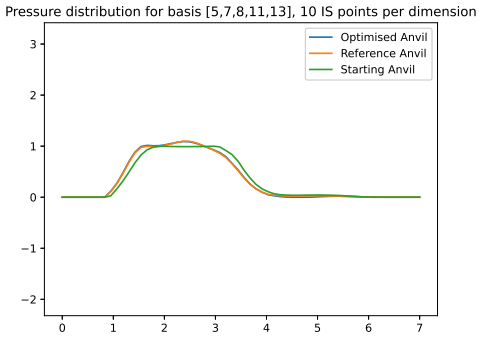


(b) The optimised anvil

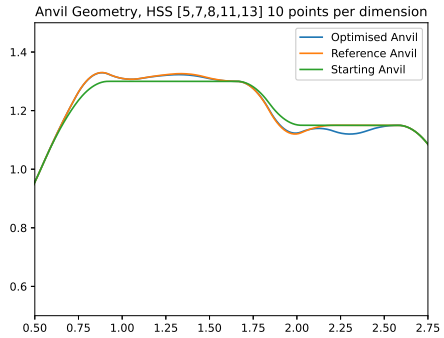
Figure 6.17: The results from running a 25 point HSS with an acquisition budget of 25 qEI points.

HSS 50, qEI 25

For this run the optimised design variables were: [0.04917, 0.01934, 0.02523, -0.04535, -0.04002] with a predicted $g_0 = 0.0004896$ and a simulated $g_0 = 0.008042$.



(a) The optimised pressure curve

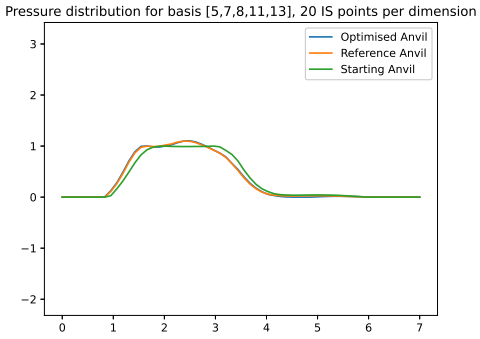


(b) The optimised anvil

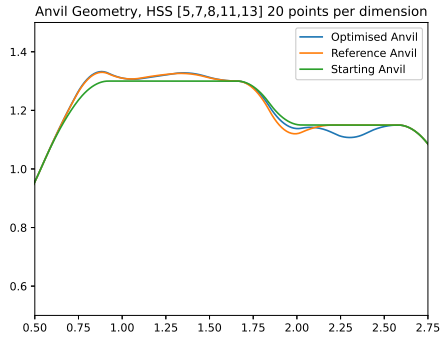
Figure 6.18: The results from running a 50 point HSS with an acquisition budget of 25 qEI points.

HSS 100, qEI 25

For this run the optimised design variables were: [0.05359, 0.01646, 0.03363, -0.02204, -0.05689] with a predicted $g_0 = 0.005781$ and a simulated $g_0 = 0.007500$.



(a) The optimised pressure curve

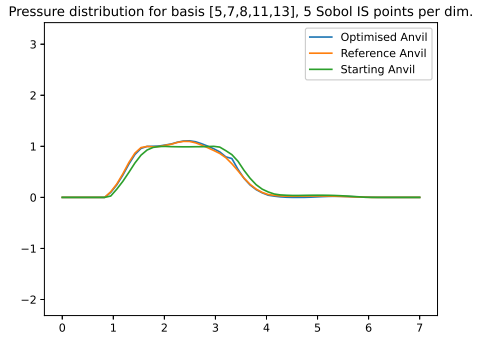


(b) The optimised anvil

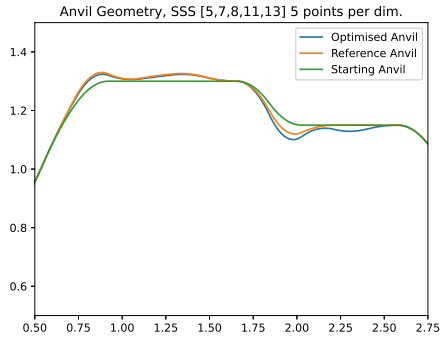
Figure 6.19: The results from running a 100 point HSS with an acquisition budget of 25 qEI points.

SSS 25, qEI 25

For this run the optimised design variables were: [0.04056, 0.01567, 0.02756, -0.07859, -0.02811] with a predicted $g_0 = 0.009844$ and a simulated $g_0 = 0.01706$.



(a) The optimised pressure curve

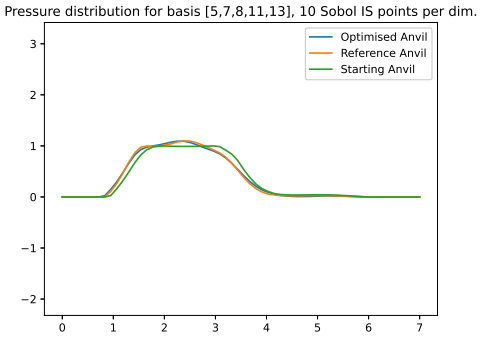


(b) The optimised anvil

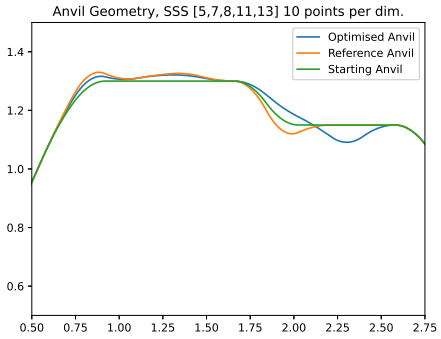
Figure 6.20: The results from running a 25 point HSS with an acquisition budget of 20 qEI points.

SSS 50, qEI 25

For this run the optimised design variables were: [0.02873, 0.01960, 0.02249, 0.05556, -0.07872] with a predicted $g_0 = -0.1205$ and a simulated $g_0 = 0.02110$.



(a) The optimised pressure curve

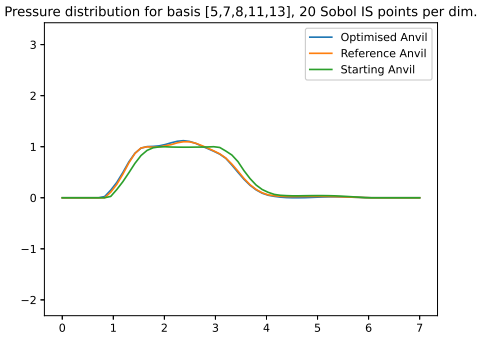


(b) The optimised anvil

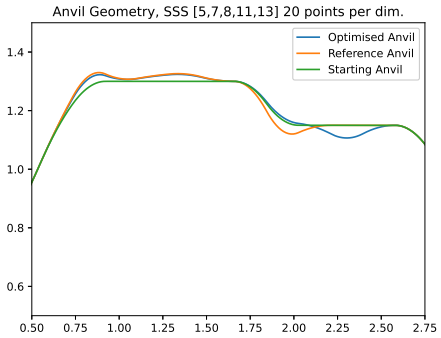
Figure 6.21: The results from running a 50 point SSS with an acquisition budget of 25 qEI points.

SSS 100, qEI 25

For this run the optimised design variables were: [0.03885, 0.01833, 0.02717, 0.01258, -0.05786] with a predicted $g_0 = 0.008064$ and a simulated $g_0 = 0.01264$.



(a) The optimised pressure curve

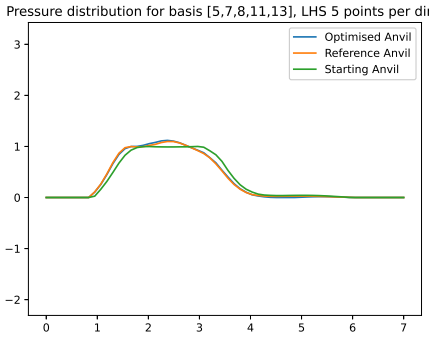


(b) The optimised anvil

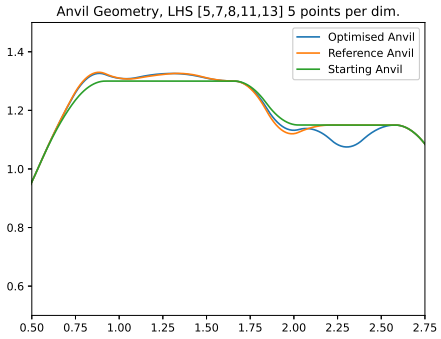
Figure 6.22: The results from running a 100 point SSS with an acquisition budget of 25 qEI points.

LHS 25, qEI 25

For this run the optimised design variables were: [0.04449, 0.02488, 0.02684, -0.03138, -0.1000] with a predicted $g_0 = -0.04151$ and a simulated $g_0 = 0.009924$



(a) The optimised pressure curve

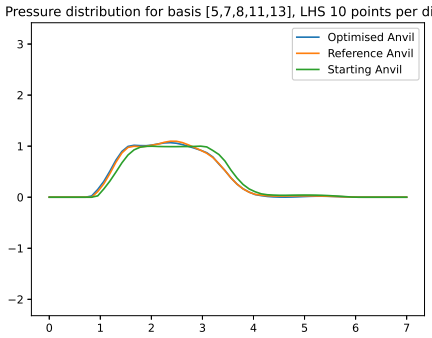


(b) The optimised anvil

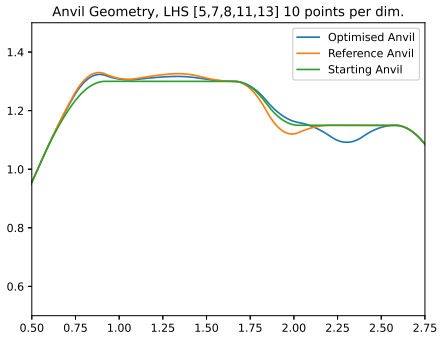
Figure 6.23: The results from running a 25 point LHS with an acquisition budget of 25 qEI points.

LHS 50, qEI 25

For this run the optimised design variables were: [0.04036, 0.01361, 0.01885, 0.01887, -0.07739] with a predicted $g_0 = -0.004775$ and a simulated $g_0 = 0.02745$



(a) The optimised pressure curve

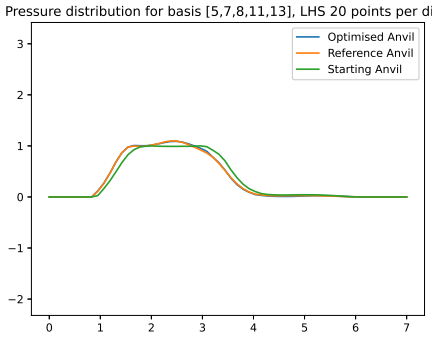


(b) The optimised anvil

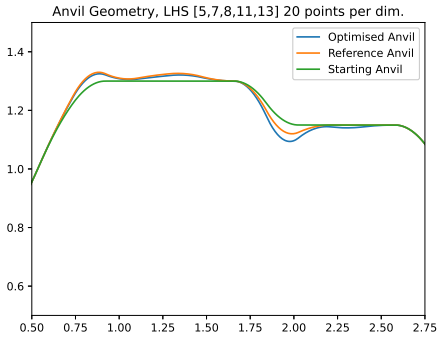
Figure 6.24: The results from running a 50 point LHS with an acquisition budget of 25 qEI points.

LHS 100, qEI 25

For this run the optimised design variables were: [0.04222, 0.01362, 0.02465, -0.08832, -0.01200] with a predicted $g_0 = 0.005492$ and a simulated $g_0 = 0.006401$



(a) The optimised pressure curve



(b) The optimised anvil

Figure 6.25: The results from running a 100 point LHS with an acquisition budget of 25 qEI points.

6.3.5 Full Anvil Optimisation

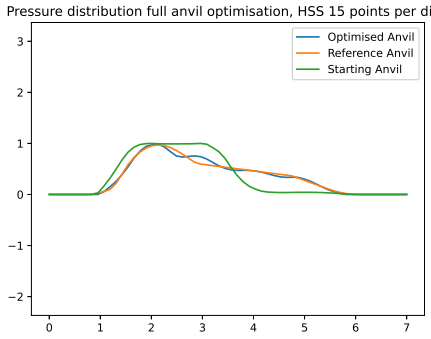
The objective function values and found values for the design variables in the full anvil optimisations can be seen in table 6.3. The following sections show results in further detail with pressure distributions and resulting geometries.

Table 6.3: The results of the optimisation experiments with known references.

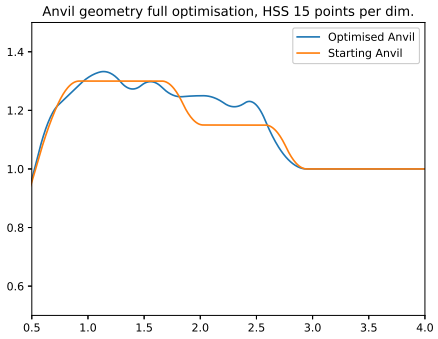
Experiment run	**Design variables α**	**Predicted g_0**	**True g_0**
HSS 180, qEI 50	$[0.03570, -0.04808, 0.02132, 0.04015, -0.04433, 0.01407, -0.05651, 0.1000, 0.1000, 0.04983, 0.1000, -0.06976]$	0.05557	0.1025
HSS 180, qEI 100	$[-0.1000, 0.004173, 0.01253, 0.03496, 0.04175, -0.06053, 0.003263, 0.1000, 0.1000, 0.09077, 0.1000, -0.04070]$	0.04796	0.05444
HSS 240, qEI 50	$[0.01113, -0.02322, -0.04506, 0.06747, -0.003629, 0.008000, -0.03882, 0.1000, 0.1000, 0.1000, 0.1000, -0.02138]$	0.1538	0.1734
HSS 240, qEI 100	$[-0.04957, -0.03113, 0.08458, -0.02550, 0.06670, -0.08187, 0.02180, 0.1000, 0.1000, 0.1000, 0.1000, -0.03632]$	0.1218	0.1502
SSS 180, qEI 50	$[0.04682, -0.03200, 0.04773, 0.01892, 0.06180, -0.1000, 0.04223, 0.1000, 0.1000, 0.1000, 0.09196, -0.1000]$	0.1274	0.3701
SSS 180, qEI 100	$[0.03583, -0.02665, 0.001963, 0.03011, 0.04487, -0.1000, 0.01743, 0.09870, 0.1000, 0.1000, 0.03063, -0.0007022]$	0.1200	0.1574

HSS 180, qEI 50

Using 15 initial sampling points per design variable and 50 qEI acquisition points in 10 batches of 5, the optimiser predicted a $g_0 = 0.05557$, with a true $g_0 = 0.1025$.



(a) The optimised pressure curve

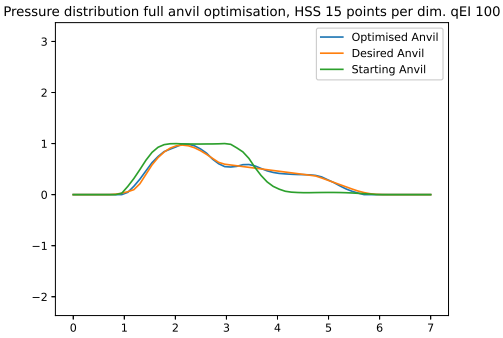


(b) The optimised anvil

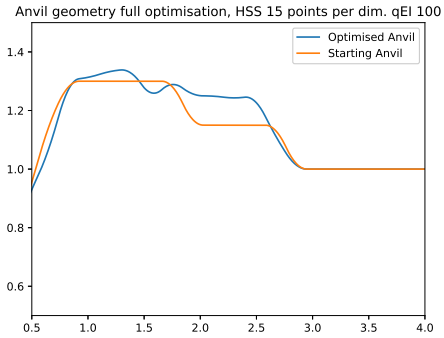
Figure 6.26: The results from running a 180 point HSS with an acquisition budget of 50 qEI points.

HSS 180, qEI 100

Using 15 initial sampling points per design variable and 100 qEI acquisition points in 10 batches of 5, the optimiser predicted a $g_0 = 0.04796$, with a true $g_0 = 0.05444$.



(a) The optimised pressure curve

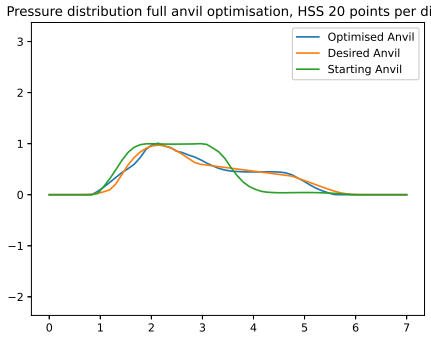


(b) The optimised anvil

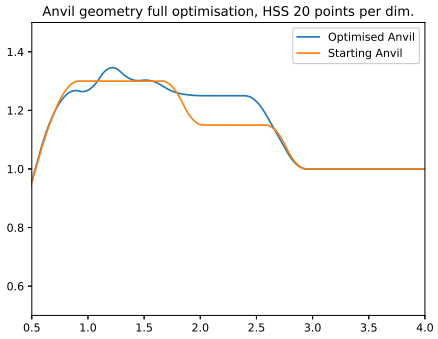
Figure 6.27: The results from running a 180 point HSS with an acquisition budget of 100 qEI points.

HSS 240, qEI 50

Using 20 initial sampling points per design variable and 50 qEI acquisition points in 10 batches of 5, the optimiser predicted a $g_0 = 0.1538$, with a true $g_0 = 0.1734$.



(a) The optimised pressure curve

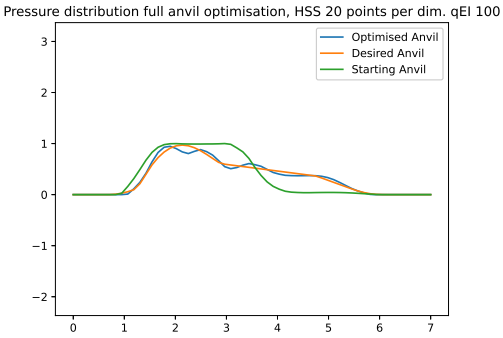


(b) The optimised anvil

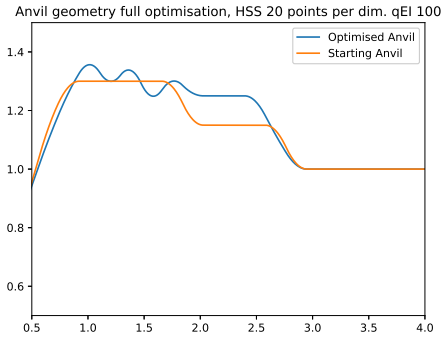
Figure 6.28: The results from running a 240 point HSS with an acquisition budget of 50 qEI points.

HSS 240, qEI 100

Using 20 initial sampling points per design variable and 100 qEI acquisition points in 10 batches of 5, the optimiser predicted a $g_0 = 0.1218$, with a true $g_0 = 0.1502$.



(a) The optimised pressure curve

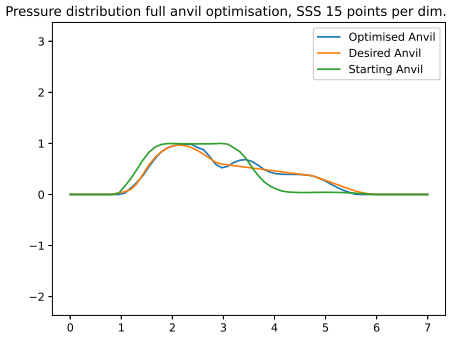


(b) The optimised anvil

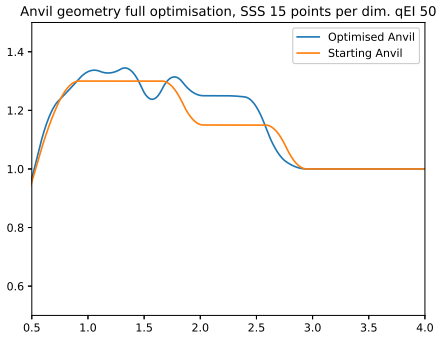
Figure 6.29: The results from running a 240 point HSS with an acquisition budget of 100 qEI points.

SSS 180, qEI 50

Using 15 initial sampling points per design variable and 50 qEI acquisition points in 10 batches of 5, the optimiser predicted a $g_0 = 0.1274$, with a true $g_0 = 0.3701$.



(a) The optimised pressure curve

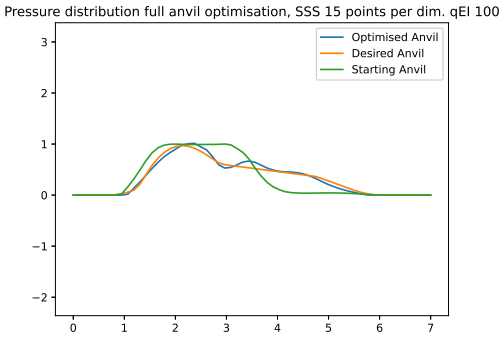


(b) The optimised anvil

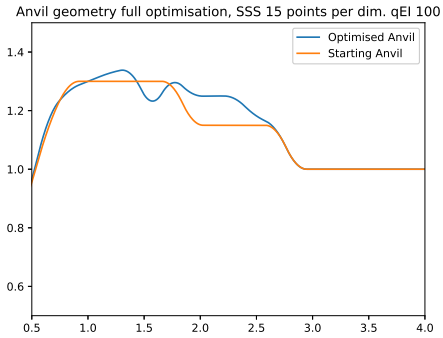
Figure 6.30: The results from running a 180 point SSS with an acquisition budget of 50 qEI points.

SSS 180, qEI 100

Using 15 initial sampling points per design variable and 100 qEI acquisition points in 10 batches of 5, the optimiser predicted a $g_0 = 0.1200$, with a true $g_0 = 0.1574$.



(a) The optimised pressure curve



(b) The optimised anvil

Figure 6.31: The results from running a 180 point SSS with an acquisition budget of 100 qEI points.

6.4 Discussion

6.4.1 Individual Experiments

1D Screening

The results of the 1D screening are somewhat expected given the locations of the basis functions tested. Basis function 3 was located on the slope towards the knife, and only affected the slope, not the main peak of the anvil. As the paperboard did not come in to any significant contact with this slope during the simulations, it comes as no surprise that the objective function is unaffected by this design variable. Basis function 7 was located on the main peak of the anvil, where the resulting pressure in the paperboard is the highest. Therefore, the results are as expected, with the objective function increasing in both directions of the screened design space. For high values, the main anvil peak gets an even higher peak, resulting in greater and more concentrated pressure. For negative values, the main peak of the anvil essentially becomes smaller and split into two as the groove gets deeper. Continuing, basis function 11 is located in such a way that is affect both main peak width, the slope to the secondary peak as well as a section of the secondary peak. It is therefore not surprising that for negative values, where basis eleven affects the main peak somewhat, the objective function is affected. For basis 13, only the secondary peak of the anvil is affected. From the results it can be seen that the secondary peak is less significant for the objective function if lowered and has a considerable significance if heightened to levels approaching the main peak. However, for the same values of design variable, the main peak is more important for the pressure distribution compared to the secondary peak. The final basis function, basis 16, only affects the secondary peak in the higher end of the screened design space, and since the secondary peak is less important for the pressure distribution, the effects of basis sixteen are essentially zero.

Performing a screening of regions and design variables provides key information that can guide the optimisation process. It allows the engineer to make decisions about what design variables to include in the optimisation process and what box constraints are interesting to explore. For the full anvil optimisation, this screening provided information which allowed certain basis functions to be excluded, saving computational time.

One Design Variable

The purpose of these runs were to test whether the methodology could improve the model in one dimension with large shape changes, and how the improvement depends on the number of acquisition points. All three runs created an anvil which improved the pressure curve, see figures 6.11-6.13. However, the run with the most number of acquisitions did not yield the best anvil. It was the second run, with 10 acquisition points that yielded the best anvil. Although it has not been shown in section 3.4.4, during testing it was discovered that, under specific circumstances, adding an additional point between two previous points has the potential to worsen the approximation between two of the points. This can occur due to the Matérn kernel being two times differentiable.

Predicted g_0 deviating from the true g_0 is to be expected, the Kriging model approximates the function values in areas where it has no points, and the input data is assumed to have some level of noise. These two factors make it inevitable that the approximation deviates from the real value. Still, we are able to obtain an anvil geometry and pressure curve that are close to the reference.

Two Design Variables

The purpose of these runs were to test if the methodology could improve the model in multiple dimensions with small shape changes, and how the improvement depends on the acquisition function. Figures 6.14-6.16 show the obtained anvils, and their respective pressure curves. The methodology was able to improve the design in all three cases, creating anvils and pressure curves

that are hard to discern from their reference in all three instances. The qEI-run had the best predicted g_0 and also the best simulated g_0, which aligns with our findings from the initial 1D testing.

An issue from the initial testing of UCB was balancing the function's exploration-coefficient β. In this case $\beta = 5$ managed to find an anvil with $\approx 30\%$ lower g_0 than UCB with $\beta = 2$ did. Finding the perfect balance of β would be difficult, as the response surface's landscape is completely unknown. It could be possible to perform a parameter study, aiming to find the best value of β for different problems, however it is likely going to yield an ambiguous answer. The initial testing, and these runs show UCB is capable of exploring the landscape, given some level of exploration added to the function, and finding an optimum close to the true optimum.

Five Design Variables

For the previous sections both the optimised pressured curves and optimised anvils have been close to the reference. However, that is no longer the case. Figures 6.17-6.25 show the optimised anvil geometry and the optimised pressure curves. The anvil geometry deviates significantly from the reference geometry in the region of the secondary peak for all five variable runs. This phenomenon can be explained from the 1D screening. Figure 6.10 show how some bases have an insignificant amount of influence on the objective function. For these bases the solution becomes ambiguous. The response surface's landscape is largely flat, and there is a loss of uniqueness in the solution.

The respective pressure curves strengthens this argument: the areas where the anvil deviates significantly, the pressure curve is nearly identical. With these runs we wanted to test if the model was able to find an optimum on a multidimensional surface, something it proved capable of. Despite there being significantly more deviation between the reference for the five variable run and the one variable run, their simulated g_0 is actually of the same magnitude.

The secondary purpose of these runs were to compare the initial sampling methods. For the runs we performed SSS performed noticeably worse, in terms of g_0, than HSS and LHS did. Though, it is not possible from our results to distinguish between the performance of HSS and LHS. Still, SSS did not perform poorly. Despite having the worst performance it was able to improve the anvil's design and the corresponding pressure curves still lie close to the reference.

Full Anvil Optimisation

As with the previous results, but particularly with these, it is important to note that it is not the anvil itself we are interested in, but the resulting pressure curve. In general, the pressure curves and g_0-values are worse than the ones from previous simulations. There are two main reasons the pressure curve distribution may be worse than before: the results suggest the shape change needed on the anvil is larger than before, and the dimensionality of the system is significantly increased. In previous tests large shape changes were attempted, but not as large as these ones. It is possible that these shape changes are simply too large for the algorithm to handle, at least in a single iteration. The fact that many design variables hit the box constraints means a second iteration would be in order. With the increased dimensionality comes increased complexity in finding a minimum, it is possible a change in internal optimisation parameters is needed to compensate for this.

To improve the optimisation algorithm, it would be interesting to experiment more with the internal optimisation parameters, e.g number of restarts for finding the minimum. Additionally, using the optimised anvil as a starting point, locking the weights where the pressure is acceptable, could be an interesting way of approaching the problem. The screening performed shows that some of the bases has little influence on g_0, a behaviour that the results show as well. Parts of the design varies significantly between runs, suggesting there is no uniqueness to the solution in those parts, as the screening suggests.

6.4.2 Summarising Discussion and Conclusion

The aim of this thesis was to create an optimisation methodology, using Bayesian optimisation, capable of improving the design of an anvil. The results show that this has been accomplished. The methodology has proven able to not only improve the design, but also get close to the sought pressure curve. Furthermore, the intention was that this document shall be able to serve as a reference for future thesis workers. To achieve this, we aimed at exploring the capabilities of the initial sampling methods, and the different acquisition functions. As EI failed early in the initial testing it was never tried on the anvil, instead qEI was primarily used. UCB also saw some testing. However, since it was deemed difficult to balance the exploration coefficient, qEI was favoured for the final testing. Despite this, UCB still performed well for the two runs it was used. To find the perfect balance of β is outside the scope of this thesis would likely take significant effort, while unlikely yielding significant results.

Furthermore, when testing the different initial sampling methods, it was not possible to discern between the performance of HSS and LHS. Although, SSS performed worse, in terms of g_0 it still performed well and the resulting pressure curve does not significantly deviate from those of HSS and LHS. It could be argued that these methods should be tested without an acquisition phase to truly gauge how they perform. Since they will never be used in a vacuum - they should always be accompanied by an acquisition phase - it was elected not to test without acquisition. The literature suggests SSS needs a significant number of sample points before it reaches its full potential, and that it only becomes superior at a dimensionality of around 10. It would be interesting to test this theory by increasing the dimensionality of the problem. It is worth noting that we ourselves have not implemented the Sobol sampling used. It was taken from BoTorch, and as such, we do not know the inner workings of it, which may be a source of error.

Looking at final stage of the optimisation, where the posterior mean of the Kriging model is optimised upon, it can be seen in table 6.2 that the sampled points sometimes had a better g_0 than what the posterior mean predicted. Three reasons for this phenomenon have been identified. Firstly, the noise added in the training of the Kriging model makes the model lose uniqueness if choosing between two points differing in a small enough magnitude. Due to this noise, the points can be assigned a posterior mean that reverses what point is actually the smallest in comparison to another.

Secondly, the phenomenon may arise when optimising the hyperparameters. For example, if the data is particularly spiky, the Kriging model can struggle to interpolate such values exactly, and instead choose hyperparameters which smooth out this feature. As a result, the model posterior mean can differ slightly from what was measured. However, the Kriging model is nevertheless an adaptive and useful surrogate model. This is something that was not investigated further as the training is done internally.

Thirdly, the phenomenon may indicate that there is a loss of uniqueness in the simulation for values that are too close to another. This could be resolved using a more rigorous simulation, including a proper mesh convergence analysis. Overall, the phenomenon should be considered a quirk arising from both BoTorch and the simulation, that was not resolved within this thesis. While it does not affect the optimisation results significantly, it is a source of error that should be taken into account when applying the method.

In conclusion, we have achieved our aim of implementing a Bayesian Optimisation framework, capable of improving the design of the anvil. Through our literature study we have outlined how an efficient Bayesian Optimisation framework may be implemented, and we have implemented a first version of Bayesian Optimisation. To make this implementation as efficient as possible more testing and fine-tuning needs to be performed. We believe that this framework can be used in the process of designing a new component. However, our results suggest that the implementation is

not yet mature enough to be used in the design process. For that, further testing and discussed improvements will have to be performed and implemented.

6.5 Future Work and Improvements

It has been shown that it is possible to use Bayesian Optimisation to improve the design of a machine component. Thus, future work should focus on fine-tuning or optimising parts of the methodology.

An idea for future improvement could be incorporating EI once an optimum is found. Instead of directly assuming the optimum is correct, it could be added to the model and a number of iterations of EI could be run locally around the found optimum. This is similar to using SBAO, which was used in the benchmarking, in the final stages.

To increase the efficiency of the method and improve the acquisition phase, including a measure of model convergence can be investigated. Such methods can draw inspiration from machine learning tactics, e.g. K-fold cross validation or bootstrapping. These validation measures test the generalisation of the model, and can give an indication if enough data has been collected. Another way of including a convergence measure is to use the change in EI and the optimum posterior mean between model refits, which could indicate a convergence. However, this would not guarantee global convergence, as it can simply be a local stagnation in the training. If successful, these improvements to the acquisition phase can allow for early stopping and not using the whole budget, which saves significant computational effort. It can also achieve the opposite, allowing more acquisitions to be performed.

Regarding the model, other types of Gaussian processes, or Kriging models with different kernels can also be used. It is highly likely that the single model that was used is the best variant possible. Other such variants can include Matérn kernels with different degrees of differentiability. For multi-objective optimisation, there exists many choices compatible with Kriging modelling.

Furthermore, it would be interesting to test how the performance of the sampling methods develop as the dimensionality increases. The thesis has put significant effort into detailing initial sampling methods, but, while the performance of them differ, the thesis has not thoroughly tested how the performance varies when dimensionality is significantly higher, dim ≥ 50. The level of noise has not been optimised, but was chosen from the initial testing. It is possible that the current level is too high for our case, and lessens the precision of the optimisation.

Finally, it is recommended to attempt an implementation of an iterative approach. As for the case with the full anvil optimisations, a solution is found which lies closer to the desired response compared to the starting point. The response is however not perfect and artefacts such as unnecessary peaks can be seen in the geometry. It would therefore be highly interesting to use the optimised solution as the initial values for a second iteration, where a more narrow design space can be used. Alternatively, a second iteration can focus on removing artefacts by locking certain basis functions that perform satisfactory, and only include problematic basis functions. This would lower the dimensionality of the problem. While it was not attempted in the project, this would also be the easiest improvement to implement and test.

Bibliography

[1] Abrash, Mohamed. "Bayesian Optimization with Applications to LPJ-GUESS". MA thesis. Lund University, 2025.

[2] Ament, Sebastian et al. *Unexpected Improvements to Expected Improvement for Bayesian Optimization*. 2025. arXiv: 2310.20708 [cs.LG]. URL: https://arxiv.org/abs/2310.20708.

[3] Archetti, Francesco and Candelieri, Antonio. *Bayesian Optimization and Data Science*. Springer Cham, 2019.

[4] Christensen, Peter W. and Klarbring, Anders. *An Introduction to Structural Optimization*. Springer, 2009. DOI: https://doi.org/10.1007/978-1-4020-8666-3. URL: https://link.springer.com/book/10.1007/978-1-4020-8666-3.

[5] Dige, Nishant and Diwekar, Urmila. "Efficient sampling algorithm for large-scale optimization under uncertainty problems". In: *Computers Chemical Engineering* 115 (2018), pp. 431–454. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2018.05.007. URL: https://www.sciencedirect.com/science/article/pii/S009813541830437X.

[6] Diwekar, Urmila M. and Kalagnanam, Jayant R. "Efficient Sampling Technique for Optimization under Uncertainty". In: *AlChe Journal* (1997).

[7] European Parliament and Council of the European Union. *Regulation (EU) 2025/40 of the European Parliament and of the Council*. OJ L. 2025. URL: https://eur-lex.europa.eu/eli/reg/2025/40/oj.

[8] Fleury, Claude. "CONLIN: An efficient dual optimizer based om convex approximation concepts". In: *Structural Optimisation* 1.2 (1989), pp. 81–89. DOI: 10.1007/BF01650999.

[9] Ginsbourger, David, Le Riche, Rodolphe, and Carraro, Laurent. *A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes*. Tech. rep. Mar. 2008. URL: https://hal.science/hal-00260579.

[10] Greif, Lucas et al. "Structured sampling strategies in Bayesian optimization: evaluation in mathematical and real-world scenarios". In: *Journal of Intelligent Manifacturing* (2025).

[11] Håkansson, Emma and Lychou, Cecilia. "Optimization and Robust Product Design of a Membrane Opening". Supervisors: Pär-Ola Jansell, Mattias Olsson, Mathias Wallin. Master's Dissertation. Lund, Sweden: Division of Solid Mechanics, Lund University, 2009.

[12] Joe, Stephen and Kuo, Frances. *Sobol sequence generator*. Accessed: 2026-03-23. University of New South Wales. URL: https://web.maths.unsw.edu.au/~fkuo/sobol/.

[13] Jones, Donald R., Schonlau, Matthias, and Welchs, William J. "Efficient Global Optimization of Expensive Black-Box Functions". In: *Journal of Global Optimization* 13 (1998).

[14] Komeilizadeh, Koushyar, Kaps, Arne, and Duddeck, Fabian. "Isovolumetric adaptations to space-filling desing of experiments". In: *Optimization and Engineering* (2022).

[15] Kucherenko, Sergei, Albrecht, Daniel, and Saltelli, Andrea. *Exploring multi-dimensional spaces: a Comparison of Latin Hypercube and Quasi Monte Carlo Sampling Techniques*. arXiv - University of Cornell (USA), 2015. URL: http://arxiv.org/abs/1505.02350.

[16] LaValle, Steven M. *Planning Algorithms*. Cambridge University Press, 2006.

[17] Leary, Stephen, Bhaskar, Atul, and Keane, Andy. "Optimal orthogonal-array-based latin hypercubes". In: *Journal of Applied Statistics* (2003).

[18] McKay, M. D., Beckman, R. J., and Conover, W. J. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2 (1979), pp. 239–245. DOI: 10.1080/00401706.1979.10489755.

[19] Persson, Johan A. *An introduction to Surrogate Modeling and Response Surface Methodology in Engineering Design Optimization*. Linköping University Electronic Press, 2026.

[20] Queipo, Nestor V. et al. "Surrogate-based analysis and optimization". In: *Progress in Aerospace Sciences* 41.1 (2005), pp. 1–28. DOI: 10.1016/j.paerosci.2005.02.001.

[21] Renardy, Marissa et al. "To Sobol or not to Sobol? The effects of sampling schemes in systems biology applications". In: *Mathematical Biosciences* 337 (2021), p. 108593. ISSN: 0025-5564. DOI: https://doi.org/10.1016/j.mbs.2021.108593. URL: https://www.sciencedirect.com/science/article/pii/S0025556421000419.

[22] Scherer, Michael, Denzer, Ralf, and Steinmann, Paul. "A fictitious energy approach for shape optimization". English. In: *International Journal for Numerical Methods in Engineering* 82 (2010), pp. 269–302. ISSN: 1097-0207. DOI: 10.1002/nme.2764.

[23] Shang, Boyang. "Space-Filling Designed Sampling from Databases". PhD thesis. Northwestern University, 2022.

[24] Sjövall, Filip. "Structural optimisation for frictionless contact". PhD thesis. Lunds Universitet, 2025.

[25] Sjövall, Filip and Wallin, Mathias. "A contact aware regularization technique for shape optimization". In: *Computational Mechanics* 76 (May 2025), pp. 945–961. DOI: 10.1007/s00466-025-02634-0.

[26] Stein, Michael. "Large Sample Properties of Simulations Using Latin Hypercube Sampling". In: *Technometrics* 29.2 (1987), pp. 143–151. DOI: 10.1080/00401706.1987.10488205.

[27] Svanberg, Krister. "The method of moving asymptotes - A new method for structural optimization". In: *International Journal for Numerical Methods in Engineering* 24 (Feb. 1987), pp. 359–373. DOI: 10.1002/nme.1620240207.

[28] Swartz, Kenneth et al. "Yet another parameter-free shape optimization method". In: *Structural and Multidisciplinary Optimization* 66 (Nov. 2023). DOI: 10.1007/s00158-023-03684-9.

[29] Wang, Jialei et al. *Parallel Bayesian Global Optimization of Expensive Functions*. 2019. arXiv: 1602.05149 [stat.ML]. URL: https://arxiv.org/abs/1602.05149.

[30] Wikipedia contributors. *Bayesian optimization — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-March-2026]. 2025. URL: https://en.wikipedia.org/w/index.php?title=Bayesian_optimization&oldid=1318172578.

[31] Wikipedia contributors. *Shape optimization*. Accessed: 2025-01-05. Wikipedia, The Free Encyclopedia. 2024. URL: https://en.wikipedia.org/wiki/Shape_optimization.

[32] Williams Christopher K. I.;Rasmussen, Carl Edward. *Gaussian processes for machine learning*. 3. print. Adaptive computation and machine learning. MIT Press, 2008. ISBN: 026218253X; 9780262182539.

Appendix A

Sensitivity Analysis

Typically, the most challenging part of the structural optimisation workflow is the sensitivity analysis, i.e. finding the derivatives of the objectives and constraints with respect to the design variables. As mentioned above, two main methods exist, direct and adjoint differentiation. In direct differentiation, all derivatives of the objective functions and constraint with respect to the design variables are found. In the adjoint method, certain derivatives can be avoided by solving an adjoint equation and acquiring an adjoint quantity, which is then used in the sensitivity [4]. To exemplify, we can study the compliance as objective function. Given that:

$$g_0 = \mathbb{F}^T \boldsymbol{u} \tag{A.1}$$

where the displacements $\boldsymbol{u}$ are given by the FE-equation $\mathbb{K}\boldsymbol{u} = \mathbb{F}$. Logically, the displacements depend on the design variables, allowing us to write our objective function as:

$$\hat{g}_0(\boldsymbol{x}) = g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x})) \tag{A.2}$$

To find the sensitivity of g_0 with respect to the design variables, we therefore need to find:

$$\frac{\partial \hat{g}_0(\boldsymbol{x})}{\partial x_i} = \frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial x_i} + \frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial x_i} \tag{A.3}$$

We can compare how each differentiation handles e.g. compliance as objective function.

Direct Differentiation

As seen in equation (A.3), three main derivatives need to be identified. Following Christensen and Klarbring (2009) we begin with [4]:

$$\frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial x_i} = \frac{\partial \mathbb{F}^T(\boldsymbol{x})}{\partial x_i} \boldsymbol{u}(\boldsymbol{x}) \tag{A.4}$$

$$\frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial \boldsymbol{u}} = \mathbb{F}^T(\boldsymbol{x}) \tag{A.5}$$

These derivatives are relatively easy, and now $\frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial x_i}$ needs to be found. Remembering that the displacements are given by the FE-equation, we can differentiate and analyse the result:

$$\frac{\partial}{\partial x_i} \left(\mathbb{K}(\boldsymbol{x}) \boldsymbol{u}(\boldsymbol{x}) = \mathbb{F}(\boldsymbol{x}) \right) \implies \frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial x_i} \boldsymbol{u}(\boldsymbol{x}) + \mathbb{K}(\boldsymbol{x}) \frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial x_i} = \frac{\partial \mathbb{F}(\boldsymbol{x})}{\partial x_i} \tag{A.6}$$

Isolating the derivative of the displacements with respect to the design variables and substituting into equation (A.3) together with the other derivatives, we get:

$$\frac{\partial \hat{g}_0(\boldsymbol{x})}{\partial x_i} = \frac{\partial \mathbb{F}^T(\boldsymbol{x})}{\partial x_i} \boldsymbol{u}(\boldsymbol{x}) + \mathbb{F}^T(\boldsymbol{x}) \mathbb{K}^{-1}(\boldsymbol{x}) \left(\frac{\partial \mathbb{F}(\boldsymbol{x})}{\partial x_i} - \frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial x_i} \boldsymbol{u}(\boldsymbol{x}) \right) \tag{A.7}$$

Utilising that $\mathbb{F}^T(\boldsymbol{x})\mathbb{K}^{-1}(\boldsymbol{x}) = \boldsymbol{u}^T(\boldsymbol{x})$, we can finally write:

$$\frac{\partial \hat{g}_0(\boldsymbol{x})}{\partial x_i} = 2\frac{\partial \mathbb{F}^T(\boldsymbol{x})}{\partial x_i}\boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{u}^T(\boldsymbol{x})\frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial x_i}\boldsymbol{u}(\boldsymbol{x}) \tag{A.8}$$

Adjoint Differentiation

Performing the sensitivity analysis of the compliance using adjoint differentiation, we begin with modifying the objective function. Again, taking inspiration from Christensen and Klarbring (2009), we begin with [4]:

$$\hat{g}_0(\boldsymbol{x}) = g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x})) + \boldsymbol{\lambda}^T(\mathbb{K}\boldsymbol{u} - \mathbb{F}) \tag{A.9}$$

The equilibrium equation is essentially adding a zero to the objective function, as the equation should fulfil $\mathbb{K}\boldsymbol{u} = \mathbb{F}$. We then differentiate equation (A.9) with respect to x_i.

$$\frac{\partial \hat{g}_0(\boldsymbol{x})}{\partial x_i} = \frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial x_i} + \frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial x_i} + \boldsymbol{\lambda}_i^T\left(\frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial x_i}\boldsymbol{u}(\boldsymbol{x}) + \mathbb{K}(\boldsymbol{x})\frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial x_i} - \frac{\partial \mathbb{F}(\boldsymbol{x})}{\partial x_i}\right) \tag{A.10}$$

Many of these terms have been previously identified, and a rearrangement of the terms such as:

$$\frac{\partial \hat{g}_0(\boldsymbol{x})}{\partial x_i} = \frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial x_i} + \boldsymbol{\lambda}_i^T\left(\frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial x_i}\boldsymbol{u}(\boldsymbol{x}) - \frac{\partial \mathbb{F}(\boldsymbol{x})}{\partial x_i}\right) + \left(\boldsymbol{\lambda}_i^T\mathbb{K}(\boldsymbol{x}) + \frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial \boldsymbol{u}}\right)\frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial x_i}, \tag{A.11}$$

and analysing the final term, we see that if $\boldsymbol{\lambda}$ is chosen wisely, the derivative of the displacements with respect to the design variables does not have to be evaluated. We therefore choose a lambda according to the adjoint equation:

$$\boldsymbol{\lambda}_i^T\mathbb{K}(\boldsymbol{x}) + \frac{\partial g_0(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))}{\partial \boldsymbol{u}} = 0 \Leftrightarrow \boldsymbol{\lambda}_i = \mathbb{K}^{-T}(\boldsymbol{x})\mathbb{F}^T(\boldsymbol{x}) \tag{A.12}$$

The $\boldsymbol{\lambda}_i$ that solves this adjoint equation, which is reminiscent of the equilibrium equation, acts as an adjoint displacement, and the derivative of the objective function is then:

$$\frac{\partial \hat{g}_0(\boldsymbol{x})}{\partial x_i} = \frac{\partial \mathbb{F}^T(\boldsymbol{x})}{\partial x_i}\boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{\lambda}_i^T\left(\frac{\partial \mathbb{K}(\boldsymbol{x})}{\partial x_i}\boldsymbol{u}(\boldsymbol{x}) - \frac{\partial \mathbb{F}(\boldsymbol{x})}{\partial x_i}\right), \quad \text{where} \quad \boldsymbol{\lambda}_i = \mathbb{K}^{-T}(\boldsymbol{x})\mathbb{F}^T(\boldsymbol{x}) \tag{A.13}$$

Appendix B

Additional tables

B.1 Benchmarking

Below are the complete result tables containing data collected in the benchmarking optimisations.

B.1.1 Design Variable Result Tables

Table B.1: 2-point optimisation design variable results over individual runs and averaged.

Sampling	**Run**	α_1	α_2	Iterations
Latin 5	Run 1	-0.125000	-0.125000	2
	Run 2	-0.125000	-0.250000	8
	Run 3	0.125000	0.250000	2
	Mean	-0.041667	-0.041667	4
Hammersley 5	Run 1	-0.250000	-0.062500	4
	Run 2	-0.250000	-0.062500	4
	Run 3	-0.250000	-0.062500	4
	Mean	-0.250000	-0.062500	4
Latin 10	Run 1	0.027778	0.083333	10
	Run 2	0.138889	0.027778	8
	Run 3	-0.250000	-0.138889	20
	Mean	-0.027778	-0.009259	13
Hammersley 10	Run 1	-0.250000	0.093750	5
	Run 2	-0.250000	0.093750	5
	Run 3	-0.250000	0.093750	5
	Mean	-0.250000	0.093750	5
Latin 20	Run 1	0.223684	-0.223684	16
	Run 2	0.013158	0.223684	35
	Run 3	0.013158	0.197368	15
	Mean	0.083333	0.065789	22
Hammersley 20	Run 1	-0.250000	0.171875	5
	Run 2	-0.250000	0.171875	5
	Run 3	-0.250000	0.171875	5
	Mean	-0.250000	0.171875	5
Latin 30	Run 1	0.112069	-0.129310	19
	Run 2	0.112069	0.008621	3
	Run 3	0.250000	-0.025862	6
	Mean	0.158046	-0.048851	9
Hammersley 30	Run 1	-0.250000	0.015625	6
	Run 2	-0.250000	0.015625	6
	Run 3	-0.250000	0.015625	6
	Mean	-0.250000	0.015625	6
Latin 40	Run 1	0.108974	0.032051	6
	Run 2	-0.250000	0.108974	4
	Run 3	-0.032051	0.224359	6
	Mean	-0.057692	0.121795	5
Hammersley 40	Run 1	-0.250000	0.210938	5
	Run 2	-0.250000	0.210938	5
	Run 3	-0.250000	0.210938	5
	Mean	-0.250000	0.210938	5
Latin 50	Run 1	0.137755	-0.178571	18
	Run 2	-0.250000	-0.229592	19
	Run 3	-0.137755	0.198980	13
	Mean	-0.083333	-0.069728	17
Hammersley 50	Run 1	-0.250000	0.101563	5
	Run 2	-0.250000	0.101563	5
	Run 3	-0.250000	0.101563	5
	Mean	-0.250000	0.101563	5
Latin 75	Run 1	-0.081081	0.229730	4
	Run 2	0.094595	0.060811	10
	Run 3	0.202703	0.135135	4
	Mean	0.072072	0.141892	6
Hammersley 75	Run 1	-0.250000	-0.160156	6
	Run 2	-0.250000	-0.160156	6
	Run 3	-0.250000	-0.160156	6
	Mean	-0.250000	-0.160156	6

Table B.2: 5-point optimisation design variable results over individual runs and averaged.

Sampling	**Run**	α_1	α_2	α_3	α_4	α_5	Iterations
Latin 20	Run 1	0.0368421	0.0368421	0.250000	0.197368	0.197368	100
	Run 2	0.005263	-0.078947	0.171053	0.250000	-0.223684	28
	Run 3	-0.068421	0.100000	0.250000	0.144736	-0.013158	100
	Mean	-0.008772	0.019298	0.223684	0.197368	-0.013158	76
Hammersley 20	Run 1	-0.100000	0.068750	-0.120370	0.170000	-0.198979	100
	Run 2	-0.100000	0.068750	-0.120370	0.170000	-0.198979	100
	Run 3	-0.100000	0.068750	-0.120370	0.170000	-0.198979	100
	Mean	-0.100000	0.068750	-0.120370	0.170000	-0.198979	100
Latin 30	Run 1	0.065517	0.079310	-0.232759	0.025862	0.146552	50
	Run 2	-0.010344	-0.010344	0.129310	-0.250000	0.060345	100
	Run 3	0.0241379	0.0931034	-0.181034	-0.250000	0.232758	58
	Mean	0.026437	0.054023	-0.094828	-0.158046	0.146552	69
Hammersley 30	Run 1	-0.100000	0.006250	0.188272	0.226000	0.066326	54
	Run 2	-0.100000	0.006250	0.188272	0.226000	0.066326	54
	Run 3	-0.100000	0.006250	0.188272	0.226000	0.066326	56
	Mean	-0.100000	0.006250	0.188272	0.226000	0.066326	55
Latin 40	Run 1	-0.053846	-0.058974	-0.057692	-0.044872	-0.032051	64
	Run 2	-0.100000	-0.028205	-0.108974	0.147436	0.250000	100
	Run 3	-0.033333	-0.069231	0.134615	0.185897	0.134165	78
	Mean	-0.062393	-0.052128	-0.010684	0.096154	0.117371	81
Hammersley 40	Run 1	-0.100000	0.084375	0.003086	0.186000	-0.158163	98
	Run 2	-0.100000	0.084375	0.003086	0.186000	-0.158163	98
	Run 3	-0.100000	0.084375	0.003086	0.186000	-0.158163	98
	Mean	-0.100000	0.084375	0.003086	0.186000	-0.158163	98
Latin 50	Run 1	0.002041	0.010204	0.117347	0.035714	-0.198979	100
	Run 2	-0.018367	-0.063265	0.015306	0.096939	-0.086735	100
	Run 3	0.091837	-0.075510	0.158163	-0.035714	0.025510	94
	Mean	0.025170	-0.042857	0.096939	0.032313	-0.086735	98
Hammersley 50	Run 1	-0.100000	0.040635	-0.182099	0.242000	0.188114	48
	Run 2	-0.100000	0.040635	-0.182099	0.242000	0.188114	48
	Run 3	-0.100000	0.040635	-0.182099	0.242000	0.188114	48
	Mean	-0.100000	0.040635	-0.182099	0.242000	0.188114	48
Latin 75	Run 1	-0.005405	0.048649	-0.168919	-0.175676	0.243243	100
	Run 2	0.083784	-0.013513	-0.243243	0.222973	0.141892	100
	Run 3	0.056757	-0.091819	-0.013514	0.081081	0.087838	88
	Mean	0.045045	-0.018894	-0.141892	0.042793	0.157658	96
Hammersley 75	Run 1	-0.100000	-0.064063	0.145062	0.238000	-0.139212	100
	Run 2	-0.100000	-0.064063	0.145062	0.238000	-0.139212	100
	Run 3	-0.100000	-0.064063	0.145062	0.238000	-0.139212	100
	Mean	-0.100000	-0.064063	0.145062	0.238000	-0.139212	100

Table B.3: 7-point optimisation design variable results over individual runs and averaged.

Sampling	**Run**	α_1	α_2	α_3	α_4	α_5	α_6	α_7	Iterations
Latin 30	Run 1	0.172414	-0.0793104	0.144827	-0.0344827	0.0482759	-0.00689655	0.0896552	63
	Run 2	0.0655172	0.0448276	-0.0482759	-0.186207	-0.00689655	-0.00689655	0.186207	96
	Run 3	0.0793103	0.0724138	0.0896552	-0.0482759	-0.0896552	-0.0620690	-0.158621	87
	Mean	0.105747	0.0126437	0.0620688	-0.0896552	-0.0160920	-0.0252874	0.0390804	82
Hammersley 30	Run 1	-0.100000	0.006250	0.150617	0.180800	0.0530612	-0.0975207	0.0721893	89
	Run 2	-0.100000	0.006250	0.150617	0.180800	0.0530612	-0.0975207	0.0721893	100
	Run 3	-0.100000	0.006250	0.150617	0.180800	0.0530612	-0.0975207	0.0721893	100
	Mean	-0.100000	0.006250	0.150617	0.180800	0.0530612	-0.0975207	0.0721893	96
Latin 40	Run 1	0.0743590	-0.0384615	-0.148718	0.117949	-0.0871795	0.0153846	-0.0256410	46
	Run 2	0.0230769	0.0230769	-0.0358974	-0.0153846	-0.0358974	-0.169231	-0.169231	85
	Run 3	0.00769231	0.00769231	0.0461539	-0.0546103	-0.0769231	-0.0358974	0.0461539	69
	Mean	0.0350427	-0.00256410	-0.0461538	0.0159847	-0.0666667	-0.0632479	-0.0495727	67
Hammersley 40	Run 1	-	-	-	-	-	-	-	Failed
	Run 2	-	-	-	-	-	-	-	Failed
	Run 3	-	-	-	-	-	-	-	Failed
	Mean	-	-	-	-	-	-	-	Failed
Latin 50	Run 1	0.0591837	0.0795918	0.175510	0.191837	-0.0448980	0.191837	-0.0367347	70
	Run 2	0.0836735	0.0836735	0.0448980	-0.0448980	-0.0857143	-0.0612245	-0.0857143	100
	Run 3	-0.0204082	-0.0183674	-0.0183674	-0.0448980	-0.126531	0.200000	0.0938776	100
	Mean	0.0408163	0.0482993	0.0673469	0.0340137	-0.0857144	0.1102042	-0.0095238	90
Hammersley 50	Run 1	-0.100000	0.0406250	-0.145679	0.193600	0.141691	-0.0314050	-0.145562	41
	Run 2	-0.100000	0.0406250	-0.145679	0.193600	0.141691	-0.0314050	-0.145562	41
	Run 3	-0.100000	0.0406250	-0.145679	0.193600	0.141691	-0.0314050	-0.145562	41
	Mean	-0.100000	0.0406250	-0.145679	0.193600	0.141691	-0.0314050	-0.145562	41
Latin 75	Run 1	0.0378378	-0.0945945	0.189189	-0.0702703	0.156757	0.124324	0.151351	100
	Run 2	-0.0189189	-0.0135135	0.0108108	0.0378378	0.151351	0.135135	-0.378378	62
	Run 3	-0.0405405	-0.0513514	0.129730	-0.183783	0.178378	-0.0216216	0.0378378	93
	Mean	-0.0072072	-0.0531531	0.1099100	-0.0720718	0.1621620	0.0792791	-0.0630631	85
Hammersley 75	Run 1	-0.100000	-0.0640625	0.116049	0.190400	-0.111370	-0.147107	0.119527	15
	Run 2	-0.100000	-0.0640625	0.116049	0.190400	-0.111370	-0.147107	0.119527	33
	Run 3	-0.100000	-0.0640625	0.116049	0.190400	-0.111370	-0.147107	0.119527	33
	Mean	-0.100000	-0.0640625	0.116049	0.190400	-0.111370	-0.147107	0.119527	27

B.1.2 Compliance and Volume Result Tables

Table B.4: 2-point optimisation compliance and volume results from all simulations, including averages over runs.

Sampling	Quantity	Run 1	Run 2	Run 3	Mean	Std
Latin 5	Compliance	1812916	1812762	1812820	1812833	77.78
	Volume	3.215930	3.216009	3.216009	3.215983	4.56e-05
Hammersley 5	Compliance	1812789	1812789	1812789	1812789	0
	Volume	3.215989	3.215989	3.215989	3.215989	0
Latin 10	Compliance	1812802	1812772	1812929	1812834	83.34
	Volume	3.216006	3.216027	3.216032	3.216022	1.38e-05
Hammersley 10	Compliance	1812789	1812789	1812789	1812789	0
	Volume	3.216013	3.216013	3.216013	3.216013	0
Latin 20	Compliance	1813056	1812961	1812440	1812819	331.64
	Volume	3.216014	3.216009	3.216062	3.216028	2.93e-05
Hammersley 20	Compliance	1812899	1812899	1812899	1812899	0
	Volume	3.216010	3.216010	3.216010	3.216010	0
Latin 30	Compliance	1812988	1812936	1813097	1813007	82.16
	Volume	3.216012	3.216025	3.216009	3.216015	8.50e-06
Hammersley 30	Compliance	1812235	1812235	1812235	1812235	0
	Volume	3.216236	3.216236	3.216236	3.216236	0
Latin 40	Compliance	1812692	1812722	1812786	1812733	48.01
	Volume	3.215944	3.216085	3.216003	3.216011	7.08e-05
Hammersley 40	Compliance	1812829	1812829	1812829	1812829	0
	Volume	3.216113	3.216113	3.216113	3.216113	0
Latin 50	Compliance	1812741	1813005	1812475	1812740	265.00
	Volume	3.216125	3.216011	3.216120	3.216085	6.44e-05
Hammersley 50	Compliance	1812851	1812851	1812851	1812851	0
	Volume	3.216081	3.216081	3.216081	3.216081	0
Latin 75	Compliance	1812641	1812450	1812672	1812588	120.23
	Volume	3.215940	3.216072	3.216121	3.216044	9.36e-05
Hammersley 75	Compliance	1812711	1812711	1812711	1812711	0
	Volume	3.216121	3.216121	3.216121	3.216121	0

Table B.5: 5-point optimisation compliance and volume results from all simulations, including averages over runs.

Sampling [method, nbr points]	Quantity	Run 1	Run 2	Run 3	Mean	Std
Latin 20	Compliance	1875901	1818043	1812797	1835580	35000
	Volume	3.214819	3.214819	3.218347	3.2	0.002
Hammersley 20	Compliance	1807648	1807648	1807648	1807648	0
	Volume	3.217826	3.217826	3.217826	3.217826	0
Latin 30	Compliance	1812423	1811950	1813189	1812521	625.24
	Volume	3.216947	3.215981	3.218906	3.217278	0.0014903278162874
Hammersley 30	Compliance	1809255	1809255	1809256	1809255	0.57
	Volume	3.216330	3.216330	3.216329	3.216330	5.77e-05
Latin 40	Compliance	1824680	1824387	1823524	1824197	600.96505722047
	Volume	3.218427	3.218688	3.218582	3.2185656666667	0.00013126436429336
Hammersley 40	Compliance	1817949	1817948	1817949	1817949	0.578
	Volume	3.218407	3.218407	3.218407	3.218407	0
Latin 50	Compliance	1811521	1839110	1806918	1819183	17410.080384651
	Volume	3.223381	3.220659	3.220125	3.2213883333333	0.0017462328977926
Hammersley 50	Compliance	1814766	1814765	1814766	1814766	0.578
	Volume	3.218828	3.218828	3.218828	3.218828	0
Latin 75	Compliance	1836134	1806184	1808360	1816892.6666667	16698.964798254
	Volume	3.222402	3.218777	3.220728	3.2206356666667	0.0018142630276047
Hammersley 75	Compliance	1814575	1806650	1819907	1812832.6666667	77.77746031681
	Volume	3.219444	3.218533	3.219154	3.2159666666667	5.7735026919084E-5

Table B.6: 7-point optimisation compliance and volume results from all simulations, including averages over runs.

Sampling [method, nbr points]	**Quantity**	**Run 1**	**Run 2**	**Run 3**	**Mean**	**Std**
Latin 30	Compliance	1811464	1809460	1834056	1818326.6666667	13658.804828144
	Volume	3.217130	3.220019	3.219140	3.218763	0.0014809378785081
Hammersley 30	Compliance	1811640	1811868	1811868	1811792	131.63586137523
	Volume	3.216045	3.216149	3.216149	3.2161143333333	6.0044427995909E-5
Latin 40	Compliance	1811923	1812845	1815649	1813472.3333333	1940.6002507815
	Volume	3.215957	3.216912	3.216193	3.216354	0.00049744044869725
Hammersley 40	Compliance	-	-	-	Failed	-
	Volume	-	-	-	Failed	-
Latin 50	Compliance	1808931	1808648	1824673	1814084	9171.4346206033
	Volume	3.219043	3.217978	3.218874	3.2186316666667	0.00057236381204036
Hammersley 50	Compliance	1809642	1809642	1809642	1809642	0
	Volume	3.216068	3.216068	3.216068	3.216068	0
Latin 75	Compliance	1812916	1812762	1812820	1812832	44.904837650797
	Volume	3.2159	3.2160	3.2160	3.2160	3.3333E-5
Hammersley 75	Compliance	1808882	1808641	1808641	1808721.3333333	139.1414148747
	Volume	3.217193	3.217061	3.217061	3.217105	7.6210235532915E-5