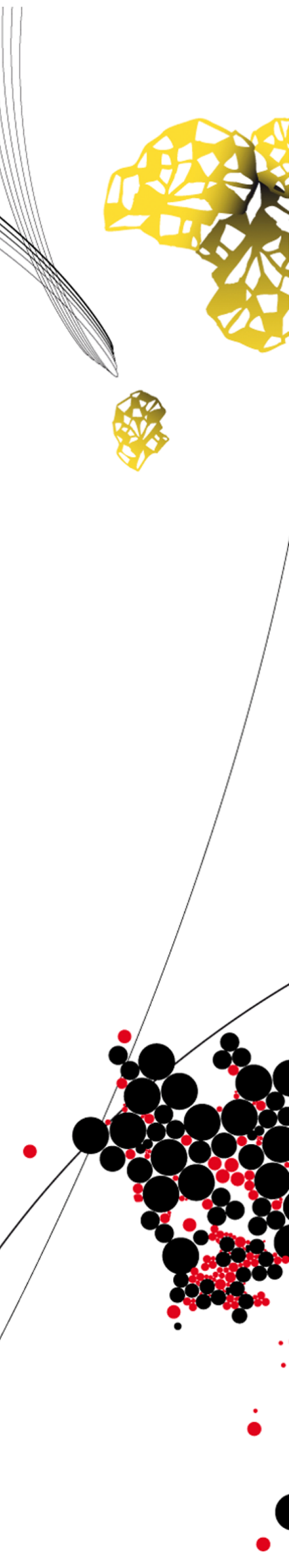


UNIVERSITY OF TWENTE.

Mechanical Engineering
Faculty of Engineering Technology

Optimization of flexure mechanisms using gradient-based methods

J. ten Hagen

M.Sc. Thesis
August 2023

Exam committee:
prof. dr. ir. A. H. van den Boogaard
dr. ir. J. Havinga
dr. ir. J. J. de Jong EngD
ir. B. Seinhorst

Nonlinear Solid Mechanics
Faculty of Engineering Technology
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Preface

Dear reader,

Welcome to reading my master's thesis titled "Optimization of flexure mechanisms using gradient-based methods". It was written for the graduation of my master's in Mechanical Engineering, at the University of Twente. This thesis is a collaboration project between the Nonlinear Solid Mechanics and Precision Engineering research chairs.

This thesis will give you a glance into the intriguing world of structural optimization. Personally, I find it intrinsically satisfying when an algorithm is finding the best possible design. There are few types of structures where optimization is more useful and interesting than for flexure mechanisms. The design of such mechanisms is inherently a trade-off between the range of motion and the dynamic performance, while the limits of the materials and manufacturability are approached.

I am thankful to have stumbled upon this graduation project. I cannot think of any other assignment that would have been nearly as interesting. During this research, I learned a lot about optimization and I became much more proficient in Matlab programming. I am certain that these skills will come in useful during my career.

This work would not have been possible without the expertise and support of my supervisors, Jos Havinga and Jan de Jong. I want to especially thank Jos Havinga for going above and beyond in supporting my work, during our weekly meetings and while proofreading my thesis. His sharp criticism and expertise in optimization and Matlab helped immensely. I would also like to thank Jan de Jong, for providing very useful insights into the modeling of flexure mechanisms. He could almost always find a gap in his busy schedule to join our meetings and was involved from the beginning to the end.

I would also like to thank Dannis Brouwer for including me in the topical meetings for the precision engineering graduate students, which provided me with feedback and a sense of the relevance of my work.

And last but not least, I would like to thank you for reading my thesis, enjoy.

Jason ten Hagen
Enschede, August 2023

Abstract

In this thesis, a method for optimizing SPACAR modeled flexure mechanisms is developed, using gradient-based algorithms. The finite differences of the objective and constraint functions were analyzed to find a suitable step size. The importance of scaling the objective function, constraints, and design space was analyzed. Using the maximum stress over the entire mechanism as constraint results in a discontinuous differentiable function, which is solved by using specific stresses as separate constraints. A similar issue can potentially occur for the objective natural frequency, when different modes catch up with each other. A reinforced cartwheel hinge was successfully and robustly optimized, using 25 design variables, showing the merit of the developed method. The proposed algorithms are sequential quadratic programming included in the MATLAB function fmincon, which is the most efficient and robust algorithm tested, and a basic quasi-Newton method implementation that can potentially handle different frequency modes in the objective function.

Contents

Chapter 1

Introduction

1.1 Background

Flexure mechanisms are designed to allow motion in certain directions while constraining motion in the other directions. This motion is achieved by elastic deformation of the flexure. Alternatively, conventional rolling and sliding mechanisms use bearings, which cause friction, wear, and backlash. This introduces unpredictability in the motion of such mechanisms. On the contrary, flexure mechanisms have highly predictable motions, making them very suitable for high-precision applications. For modeling these mechanisms, the specialized software package SPACAR [1] is used. This software is based on non-linear finite element theory for multi-degree of freedom mechanisms. The calculation time of a SPACAR model is typically twenty times faster compared to a conventional finite element model [2].

In engineering, product performance is pushed to its boundaries by careful design, such that certain performance is enhanced while accounting for all requirements that must be met. The process of finding the best performing design among all design options is called optimization. In structural design, performance can for example be defined as low mass, high stiffness, or high eigen frequencies. The requirements can be related to internal stresses, stiffnesses, or product dimensions. In optimization theory, the performance target is called the objective, and the requirements are called constraints. When having a model of the product to compute objective and constraint values for different designs, it is possible to perform computational optimization, which is an automated procedure for finding an optimal design.

Flexure mechanisms generally suffer from poor support stiffness in non-compliant directions, especially in a large stroke deflected position. Design choices in the mechanism can greatly influence the support stiffness and parasitic eigenfrequencies. Even in the simplest flexure mechanisms, multiple design choices have to be made which can influence the performance. The mechanism is also bound by constraints, such as the maximum allowable stress. Design guidelines are generally not sufficient to find the best possible design. Therefore it is often desirable to optimize a design.

Optimization of SPACAR models is mostly done using the Nelder-Mead algorithm [2, 3, 4, 5]. This is a gradient-free algorithm. It was stated that this algorithm was used, as opposed to the more efficient gradient-based algorithms, because acquiring derivatives can be difficult. The Nelder-Mead algorithm has been able to find the optimum of many flexure mechanisms, although the algorithm has to run many times from different starting points, since the algorithm does not always converge to the global optimum. There is no hard limit to the number of design variables that can be optimized with the Nelder-Mead algorithm, but 6-8 variables is generally seen as the maximum for a reasonable optimization time.

In theory, gradient-based algorithms are much more efficient, which could allow for optimization of a design with a significantly higher number of design variables. Gradient-based algorithms are also more robust, since the Nelder-Mead algorithm is known to converge to a non-stationary point in some cases

[6]. For these reasons, it can be very valuable to find a robust method of gradient-based optimization for flexure mechanisms. Although earlier attempts at reliable optimization of SPACAR models with gradient-based algorithms failed, no extensive effort was done to locate and solve the difficulties.

1.2 Research objective

The main objective of this research is to develop an optimization method for designs modeled in SPACAR, accommodating a larger number of design variables than currently used methods allow. This method should allow the optimization of a wide range of flexure mechanisms.

Part of this objective is to select or develop a suitable algorithm. The algorithms considered in this research are limited to gradient-based algorithms. The other part of the main objective is to develop a robust method of handling the optimization problem. The most common issues and obstacles in the optimization process should be identified and if possible solved.

1.3 Outline

In chapter 2, the theory of gradient-based optimization is discussed. This chapter also includes some background information about previous optimization efforts regarding SPACAR models. In chapter 3, the methods used in this research are presented, based on the theory discussed in chapter 2. In chapter 4 the different aspects of the optimization process are introduced, including a parameter sweep and a finite difference analysis. Furthermore the methods of optimization used in this research are demonstrated, using a simple cross-hinge design as example. The performance of the different optimization methods is also analyzed in this chapter. In chapter 5, two problems that introduce discontinuous differentiable objective functions or constraints are shown. For the stress constraint, a solution is presented to overcome this issue. In chapter 6, the developed method is tested on a cartwheel hinge with reinforcements. Different versions are successfully optimized with up to 25 parameters. Finally, in chapter 7, the results are discussed and conclusions are formulated.

Chapter 2

Optimization algorithms

2.1 Optimization of flexure mechanisms

An optimization problem is generally formulated as follows:

Minimize:

$$f(x) \tag{2.1}$$

Subject to:

$$\begin{aligned} &g_i(x) \leq 0, && i = 1, ..., m \\ &h_j(x) = 0, && j = 1, ..., n \\ &{x_k}^{lb} \leq x_k \leq {x_k}^{ub}, && k = 1, ..., p \end{aligned}$$

In which:

- $f(x)$ is the function to be minimized, called the objective function.
- x is a vector of all design variables, which are allowed to change within the bounds. Usually, there is a lower bound (lb) and an upper bound (ub).
- $g_i(x)$ are inequality constraints.
- $h_i(x)$ are equality constraints.
- $m \geq 0$, $n \geq 0$ and $p \geq 1$.

The objective function is conventionally always minimized. If a function has to be maximized, the negative or inverse of this function can serve as the objective function. In this work, only inequality constraints and bounds are used as constraints. The maximum allowable stress in the leaf springs under load is an example of such an inequality constraint.

Scaling of the objective function, the design variables, and the constraint functions is an important consideration in optimization. Optimization algorithms generally work best if all the design variables, the objective function, and the constraint functions are in the same order of magnitude. The design variables can be normalized from zero to one. The constraint functions are conventionally normalized from minus one to zero, where a positive value indicates that the constraint was exceeded. The objective function can be scaled by calculating the objective function value at the starting point and dividing the objective function by this value. Some objective functions span many orders of magnitude within the design space. In that case, it can be effective to take the log or square root of the objective function apart from scaling. Note that the location of the minimum does not change by scaling the objective function.

2.2 Literature overview

In this section, the relevant literature on the optimization of flexure mechanisms is reviewed. Previous optimization efforts of flexure mechanisms modeled in SPACAR are the main focus, but some other optimization work is reviewed as well.

2.2.1 SPACAR design optimization

Successful design optimizations of SPACAR models have been performed. Wiersma et al [2, 3] optimized a number of large stroke hinges of various types. The objective function was the inverse of the second eigenfrequency. The actuation moment and maximum stress were taken as constraints. For these optimizations, the Nelder-Mead simplex optimization algorithm [7] was used. This is an algorithm for unconstrained problems, but the algorithm was modified so that parameter sets that violate a constraint are not admissible. Not much information is given about the performance of the optimization algorithm. However, it was stated that the optimizations were successful and that multiple runs from different starting parameter sets converged at least more than once to the same optimum. This implies that it is likely that the global optimum was found. The number of design variables in these hinges varied from four to eight.

Naves et al [4, 5] also performed optimization of similar large stroke hinges, using the Nelder-Mead optimization algorithm. In the first paper [4], a few changes compared to the earlier work of Wiersma [2], were discussed. Instead of inadmissible parameter vectors outside of the bounds and constraints, a penalty function is added. If a point of the simplex falls in the infeasible space, the function value is multiplied by a third order penalty, ensuring that the algorithm will not converge to an infeasible location. A higher order penalty is also possible but is more likely to cause numerical issues. Another strategy that was added is constraint interpolation between a feasible and infeasible point, to estimate where the infeasible area starts without requiring extra function evaluations.

The performance of the optimization algorithms was discussed more extensively compared to Wiersma's work. The algorithm was tested on a three flexure cross hinges, with four design variables. A total of 50 shape optimizations were performed, starting from 50 randomly generated parameter sets. The optimization was terminated each time when the first eigenfrequency of subsequent iterations differed less than 0.5%. This is a rather large threshold, which could result in premature termination of the optimization process. On average 135 function evaluations were performed before termination. From the 50 runs, 52% converged within 5% of the best solution found and 24% converged within 1% of the best solution.

It was claimed that one of the reasons for not finding the optimum each time was the presence of local optima in the search domain, but this claim is not further substantiated. With a loose termination criterion, it is possible that certain runs were prematurely terminated while converging to the global optimum. Furthermore, the possibility of failure of the Nelder-Mead algorithm is completely ignored. Multiple studies have shown the lack of robustness of the Nelder-Mead algorithm, especially in problems with more than a few variables. McKinnon [8] showed a group of convex and continuous differentiable functions with two variables for which the Nelder-Mead algorithm converges to a non-stationary point. Torczon [6] studied the convergence of the Nelder-Mead algorithm on well known functions such as multi-dimensional Rosenbrock functions and concluded a discerning lack of robustness. In Torczon's research, the search direction went orthogonal to the gradient for larger dimensional functions, which cannot result in an optimum. Note that these issues exist without the added complexity of constraints and barrier functions.

Wiersma and Naves [2, 4] provide no reason for not using gradient-based optimization other than that derivatives might be difficult to obtain.

2.2.2 Other flexure mechanism optimizations

Kuresangsai et al. [9] developed a kinematic modeling and optimization technique for flexure-jointed planar mechanisms. One of the optimized models was a linear actuator using two parallel Hoeken's linkages connected to the end effector, which should be able to move ±10 mm in the x-direction. The goal of the optimization was to minimize the displacement in the y-direction at the maximum stroke length to both sides. The design variables were the positions and orientations of all the leaf springs. Only bounds were used as constraints. For the optimization algorithm, a gradient-based method was used. The gradients were directly taken from the model. The result was a very efficient optimization process. However, the method presented is limited to kinematic design optimization, for planar mechanisms.

Lobontiu et al. [10] presented an analytical model of flexure-based displacement amplification mechanisms, which were optimized for maximum stiffness and displacement amplification. Twelve design variables were used, including the radius of corner-fillets of the flexures, resulting in notched hinges. The optimization algorithm used was a direct application of the Lagrange's function and the Kuhn-Tucker conditions, which is also gradient-based optimization. Bounds of the design variables and some combined geometry bounds formed all the constraints used.

2.3 Numerical derivatives

For gradient-based optimization algorithms, derivatives of the objective function with respect to the design variables are required. The vector of first order derivatives is called the gradient. In some cases, the derivatives can be calculated analytically, but in many cases, the derivatives have to be calculated numerically. These derivatives are determined by using finite differences. Forward differences and central differences are the most used methods. The forward difference calculation requires $N + 1$ function evaluations, where N is the number of design variables. The central difference method is more accurate but requires $2N + 1$ function evaluations. Since function evaluations can be costly, forward differences are mainly used in optimization. The forward difference to one of the design variables can be calculated with equation 2.2, in which x is the design variable and h is the step of this design variable.

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{2.2}$$

The step size h is not arbitrary. For a smooth function, the smaller the step size, the more accurate the derivative is. With an increase of step size, the truncation error increases. However, the output of finite element software such as SPACAR is generally not completely smooth. There is usually some numerical noise on the output. If the step size is too small, this noise results in a large error. The range of acceptable step sizes is problem dependent. The presence and severity of noise on the function can vary widely by differences in software or settings.

Second order derivatives can be approximated by finite differences as well. Similar to first order finite differences, it is also possible to use forward differences, backward differences, and central differences. Forward differences require fewer function evaluations, but the truncation error is of first order, while for central differences the truncation error is of second order [11]. For second order finite differences noise on the function has more impact compared to first order finite differences, thus requiring a larger step size. To minimize the impact of the truncation error caused by using a larger step size, central differences will be used for second order finite differences in this work.

The full matrix of second order derivatives is referred to as the Hessian matrix. The Hessian is a symmetrical matrix, so only half of the second order mixed partial derivatives need to be calculated. The second order partial derivatives on the diagonal are calculated with equations 2.3 and the mixed partial derivatives, which are off-diagonal are calculated with equation 2.4.

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f(x+h_x) - 2f(x) + f(x-h_x)}{h_x^2} \tag{2.3}$$

$$\frac{\partial^2 f}{\partial xy} \approx \frac{f(x+h_x, y+h_y) - f(x+h_x, y-h_y) - f(x-h_x, y+h_y) + f(x-h_x, y-h_y)}{4h_x h_y} \tag{2.4}$$

To calculate the full Hessian matrix, many function evaluations are required. Each diagonal component requires two extra function evaluations and each off-diagonal component requires four function evaluations. If n is the number of design variables, equations 2.5 and 2.6 gives the required number of function evaluations N to calculate the gradient and Hessian matrix respectively.

$$N_{gradient} = n + 1 \tag{2.5}$$

$$N_{Hessian} = 2n^2 + 1 \tag{2.6}$$

2.4 Optimization algorithms

This section contains short theoretical information about the optimization algorithms used in this work. First, the Newton's method and a quasi-Newton method using a BFGS Hessian approximation are discussed. Next, the relevant algorithms available within the fmincon function which is part of the MATLAB optimization toolbox, are discussed. These algorithms are the interior-point, SQP, and active-set algorithms. Finally, the Nelder-Mead algorithm is shortly discussed, as it will be used as reference algorithm.

2.4.1 Newton's method

Newton's method is based on a local second order approximation of the objective and constraint functions. The gradient and Hessian is required at each iteration to estimate the objective function and constraint values around the current location in the design space. The first and second order derivatives give information about the direction and curvature of the functions. This gives not only the search direction, but also a good indication of the step length, since the curvature predicts the location where the gradient of the function becomes zero. At each iteration, second order Taylor expansions, shown in equations 2.7 are done around the current design point, to a acquire a quadratic function for the objective function and all constraints.

$$f(x+s) \approx f(x) + \nabla f(x)^\mathsf{T} s + \frac{1}{2} s^\mathsf{T} H(x) s, \tag{2.7}$$

Where s is the step vector from the current design point and H is the Hessian matrix. The optimization problem from Eq. 2.1 is now approximated with a set of quadratic functions, which can be solved using any quadratic programming algorithm.
The main disadvantage of the Newton's method is the requirement of a fully calculated Hessian. The number of function evaluations required for generating the Hessian using finite differences grows exponentially with the number of design variables. Therefore, the Newton's method is generally not used for problems with many design variables. To circumvent this problem while maintaining second order information, quasi-Newton methods were developed, which are discussed in the next section.

2.4.2 Quasi-Newton method method

In a quasi-Newton method, the Hessian matrix is not calculated, but instead approximated using first order derivatives of consecutive iterations. Initially, an arbitrary Hessian is chosen, which is updated each

iteration. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) update is considered the most effective quasi-Newton update [11]. The BFGS update formula is shown in equation 2.8, where $\tilde{H}$ is the approximated Hessian. In this equation, k indicates the previous location and $k+1$ indicates the current location.

$$\begin{aligned} s_k &= x_{k+1} - x_k \\ y_k &= \nabla f(x)_{k+1} - \nabla f(x)_k \\ \tilde{H}_{k+1} &= \tilde{H}_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{\tilde{H}_k s_k s_k^\top \tilde{H}_k}{s_k^\top \tilde{H}_k s_k} \end{aligned} \tag{2.8}$$

For the first iteration, an arbitrary positive definite Hessian is chosen. The initial Hessian used in this work is:

$$\tilde{H}_0 = \frac{1}{\|\nabla f_0\|} I \tag{2.9}$$

In which I is the identity matrix. The BFGS method or close variants thereof are also used in the MATLAB fmincon function.

2.4.3 Fmincon algorithms

Fmincon is a MATLAB function for the optimization of nonlinear constrained problems. The function itself is not an optimization algorithm, however, a number of algorithms can be chosen to use within fmincon. In the previously discussed methods, constrained optimality was not yet discussed, but applied in the fmincon step while solving the quadratic approximation. To solve a constrained optimality problem, the first order optimality condition is modified to include the constraints. Instead of just the objective function, the Lagrangian function, shown in equation 2.10 is introduced.

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_{g,i} \cdot g_i(x) + \sum_{i=1}^{m} \lambda_{h,i} \cdot h_i(x) \tag{2.10}$$

Instead of the requirement that the gradient of this function must be zero, the Karush-Kuhn-Tucker (KKT) conditions, see equation 2.11, must be met.

$$\begin{aligned} &\nabla_x \mathcal{L}(x, \lambda) = 0 \\ &\lambda_{g,i} g_i(x) = 0 \quad \forall i \\ &\begin{cases} g(x) \leq 0 \\ h(x) = 0 \\ \lambda_{g,i} \geq 0 \end{cases} \end{aligned} \tag{2.11}$$

The three relevant options are the interior-point algorithm, the sequential quadratic programming (SQP) algorithm, and the active-set algorithm. These algorithms will be briefly discussed in the next sections.

The SQP and active-set algorithms

The SQP and active-set algorithms are very similar, where the SQP algorithm has the same base functionality as the active-set algorithm, but with some added benefits. The active-set algorithm checks at each iteration if the constraints are active or inactive, and solves the KKT conditions shown in equation

2.11, including the active constraints. The SQP algorithm is considered the most efficient nonlinear optimization algorithm for a wide range of problems [12].
The active-set and SQP algorithm resembles the quasi-Newton discussed before. At each step, the function is transformed into a quadratic sub-problem. First order derivatives are numerically determined and the Hessian is approximated using the BFGS or similar method.

The SQP algorithm in fmincon has the following improvements over the active-set algorithm [12]:

- The algorithm respects the bounds, including the finite difference steps. If a starting point outside the bounds is tried, the starting point is moved to the bounds.
- If an attempt fails, because one of the functions returns an invalid number such as NaN or Inf, the algorithm attempts another step.
- SQP uses a more efficient set of linear algebra routines to solve the quadratic programming sub-problem.
- The SQP deals slightly differently with situations where the constraints are not satisfied.

Interior-point

In the Interior-point algorithm, a logarithmic barrier function is added to the objective function. When the constraints are approached or breached, this barrier function increases the objective function value. Each step of the Interior-Point method stays within the feasible area. The initial point is normally also required to be in the feasible area. However, in fmincon it is not necessary to start in the feasible area, since the function will first search for the feasible area before conventionally using the interior-point algorithm.

The algorithm uses two different sorts of steps. First, it attempts a direct step. This step tries to solve the KKT equations. This is called a Newtons step. If this is not possible, a conjugate gradient step is done. A quadratic approximation within a trust region, subject to linearized constraints is used. The trust region has a radius R and is there to limit the step size of all design variables.

2.4.4 Nelder-Mead

The Nelder-Mead method is mainly used in this research to compare with the gradient-based methods. It is one of the most used direct-search methods, meaning that no derivatives are required. For a n-dimensional problem, the algorithm starts with a n+1 node simplex. At each iteration, a new node is chosen and an old node is discarded. This is done in such a manner that the function value decreases. Depending on the function values at the nodes, one of a few different steps can be taken at each iteration. Since the Nelder-Mead algorithm has been clearly described in many papers and is not the focus of this research, this algorithm is not described in more detail. For a detailed explanation of the algorithm as used in the MATLAB function fminsearch, see the work of Lagarias et al [13], whose algorithm is actually implemented in fminsearch.

2.5 Local minima

The gradient-based optimization algorithms in this research and the Nelder-Mead algorithm are local search methods. These algorithms will try to find an optimum, but there is no guarantee or even indication that the minimum found is actually the global minimum. The only method of finding out if the minimum found is the global minimum, is performing the optimization multiple times from different starting points. If all runs converge to the same point, it is very likely that this point is the global and only minimum. If different points are found, but a sufficient amount of runs are performed and each point is found multiple times, it is very likely that there are multiple local minima and that the best minimum is the global minimum.

2.5.1 Optimality conditions

When an optimization algorithm converges to a certain point, it is not guaranteed to be a local minimum. The formal method to ensure a minimum is found, is to check the first and second order optimality conditions. For unconstrained problems, the first optimality condition is met if the gradient of the objective is zero. For constrained problems, the gradient in feasible directions has to be equal to or greater than zero. The first order optimality conditions are known as the Karush-Kuhn-Tucker conditions [11]. For problems with only inequality constraints, these conditions are:

$$\begin{aligned} \nabla f + J_g^\top \lambda &= 0 \\ \lambda &\geq 0. \end{aligned} \tag{2.12}$$

In which λ are the Lagrangian multipliers and J_g is the Jacobian of the active constraints and bounds. The subspace of feasible direction p is determined by taking the nullspace of J_g. With all this information, the first order optimality condition is met if:

$$p^\top \nabla f = 0 \tag{2.13}$$

The second optimality condition is met if the Hessian matrix of the objective function at the optimum is positive definite. An example of a situation where the first optimality condition is met, but the second condition is not met, is a saddle point in a two dimensional problem. The second optimality condition can be formulated as follows:

$$p^\top H_{\mathcal{L}} p > 0 \tag{2.14}$$

For all p such that:

$$J_g p \leq 0 \tag{2.15}$$

In which $H_{\mathcal{L}}$ is the Hessian of the Lagrangian:

$$H_{\mathcal{L}} = H + \lambda H_g \tag{2.16}$$

For the optimality condition check, the gradient and Hessian matrix of the objective function are required. These have to be determined with finite differences for SPACAR optimization problems. Calculating the Hessian can be very costly for larger problems. If there is no time or computational means to calculate the full Hessian, just checking the first optimality condition is already useful. Another problem of a numerical optimality condition check is numerical noise and inaccuracies. The gradient is not exactly zero, active constraints are not exactly zero and the Hessian is not completely accurate. Therefore margins are required. If the outcome of the optimality condition check is uncertain, a parameter sweep can be performed near the optimum in the feasible directions p, which can support the conclusion if optimality was reached or not.

Chapter 3

Methods

In this chapter, the methods used in this research are introduced. First, some problem analysis tools are introduced, which are used to identify the relation of the design variables to the objective and constraint functions and to find an appropriate finite difference step size. The implementations of the algorithms introduced in the previous chapter are further presented. All the optimization runs were performed in MATLAB R2021a. All model evaluations were performed in SPACAR, using SPACAR Light as interface between MATLAB and SPACAR.

3.1 Problem analysis

In the literature overview, it became clear that little is known about the performance of gradient-based optimization of flexure mechanisms modeled in SPACAR. Some successful gradient-based optimizations were performed on flexure mechanisms in general, which is promising, but these were fully analytical problems, with relatively simple constraints. For now, it is not possible to extract analytical derivatives from SPACAR. In this work, the optimization will be mostly considered as black-box optimization. This is generally the case for optimization when simulations are run as function evaluations. In black-box optimization, the output of the model from the user input can be observed, but the model is not available in analytical form.

In order to develop an optimization method, a test design is required. In this work, flexure hinges with varying complexities are used for optimization. These types of mechanisms are suitable subjects for optimization as demonstrated in earlier works. The aim of the optimizations will be to minimize the first eigenfrequency. A rotational stroke in the compliant direction of the hinges is modeled, such that the first eigenfrequency corresponds to the first parasitic mode. Von Mises stress and actuation moment are both used as constraints. Throughout this work, the first eigenfrequency is always the first parasitic mode.

The first steps are to analyze the outputs of the model for various inputs. The design space is explored using the parameter sweep technique. The accuracy and viability of acquiring derivatives with finite differences is also analyzed. Using this information, optimization methods are developed. The last section of this chapter gives an overview of the order in which the results in the next chapters are presented.

3.1.1 Parameter sweep

Before optimizing a problem it can be very useful to explore the design space first. A parameter sweep is an excellent tool for this purpose. Since it is impossible to plot a multidimensional design space, separate plots are made, where only one variable is changed at a time. From a certain starting point, one variable at a time is varied from the lower bound to the upper bound. The resulting plots can give a lot of information, which can help select a suitable optimization method. The objective function and any other relevant output of the model, such as constraints, can be plotted in this sweep. Note that a parameter sweep never gives a full picture of the entire design space, so it does not guarantee to show every problem,

but it still gives a lot of insight.

From the parameter sweep, different aspects of the objective and constraint functions can be observed. Missing data points due to failed model evaluations must be dealt with, either by improving the model and its convergence or by ensuring that the optimization algorithm can deal with unsuccessful function evaluations. It can be observed how the different parameters have no effect, a linear effect, or a nonlinear effect on the different model outputs. Furthermore, it can be observed whether these effects depend on other parameter settings, which means that there are interactions between the parameters. Also, discontinuities and discontinuities of the derivatives may be identified from the parameter sweep. Finally, it can be observed whether the output is noisy, which is relevant for determining the derivatives of the function. All these aforementioned aspects are relevant for the optimization and may be used to reformulate the optimization problem, define the required improvements of the model, and select an adequate optimization algorithm.

3.1.2 Finite difference step size sweep

For accurate derivatives, the step size is very important, as discussed in section 2.3. The noise on the output of the model determines how small of a step can be taken. The noise relevant for numerical derivatives is of a much smaller scale than can be detected with a regular parameter sweep over the entire design space. What can be done to find good step sizes is a step size sweep. From a random location in the design space, the forward difference of the objective function with respect to one of the design variables is calculated for a series of different step sizes. For increasingly small step sizes, the noise becomes dominant and the forward difference is inaccurate. For larger step sizes the truncation error dominates. In that case, the forward difference is inaccurate because of non-linearity in the output. Visualized in a graph the domain where the error is acceptable can be found.

3.2 Optimization methods

In this section, the specific implementations of the optimization methods used in this work are discussed. First, some general strategies are discussed. To test the performance of the algorithms, a multi-start is required. The different runs are started from different points spread through the design space. Generating these starting points can be done with any design of experiment technique. In this research the Latin hypercube method is used, to ensure a good spread through the design space. The random number generator in MATLAB was set to default, for repeatability. A multi-start should also be used, to make sure the global optimum is found in a potentially multi-modal problem.

Scaling and normalizing of the objective and constraint functions and the design space can have a large influence on the convergence rate of the optimizing algorithms. This is further explored in section 4.3.1. In this research, the design variables are normalized between 0 and 1, and the constraints are normalized between -1 and 0. The objective function was defined as $-f_1(x)$ in chapter 4 and as $-f_1(x)/f_1(x_0)$ in chapters 5 and 6.

The units within SPACAR were set to meters in chapter 4, and to millimeters in chapters 5 and 6. Millimeters are preferable, because in meters several numbers can get very small, causing numerical issues. Furthermore, the default of ten load steps for the applied rotation is always used. These are the steps from the equilibrium position, or the initial rotation, to the final position. For each step, the frequency, stresses, and other outputs are calculated. The stress is generally highest at the maximum rotation, while the stiffnesses and eigenfrequencies are generally lowest at the maximum rotation.

No termination criteria are implemented in the Newton's method and quasi-Newton method, instead the optimization is terminated upon reaching the iteration limit. For the fmincon algorithm, the termination criteria are set extremely strict. This way, the convergence comparison of the algorithms is not influenced by different termination criteria. In chapter 6, the default termination criteria of fmincon were

used, which are still extremely strict, because stricter termination criteria caused instability near the optimum, for the problems with many design variables.

For all the fmincon algorithms, the finite difference step size was set to 10^{-6}, instead of the default setting of approximately 10^{-8}.

3.2.1 Newton's method implementation

In each iteration of the Newton's method, the gradient and Hessian matrix of the objective function and constraint functions are determined, using finite difference step sizes of 10^{-3} in the normalized domain. The quadratic sub-problem calculated with equation 2.7, is then solved using fmincon, with the SQP algorithm.

3.2.2 Quasi-Newton method implementation

The implementation of the quasi-Newton method is similar to the Newton's method, except the Hessians of the objective and constraint functions, which are now approximated using the BFGS update equation shown in equation 2.8. A trust-region was used in this method. This limits the step length between iterations, which was necessary for stabilizing the convergence in the first few iterations. The trust-region is implemented by setting an extra constraint, which limits the norm of the step, to 0.2 in the normalized design space. The lack of a convergence criteria in the quasi-Newton method, caused the BFGS updates to become inaccurate when the step length between consecutive iterations approached zero. To counter this, the Hessians are only updated when the norm of the step vector s is larger than 10^{-16}. The finite difference step of 10^{-4} is used.

The quadratic sub-problem for this method is also solved using fmincon with the SQP algorithm. In this method, the accuracy of the solution of the quadratic sub-problem has a large influence on the performance of the method. Very strict termination criteria proved to be necessary. The StepTolerance and ConstraintTolerance, see MATLAB fmincon documentation [14], were both set at 10^{-12}, instead of the default 10^{-6}.

3.2.3 Nelder-Mead implementation

The Nelder-Mead algorithm in an unconstrained optimization tool. To take the constraints and bounds into account, a soft penalty is added to the objective function, when a bound or constraint is violated. Equation 3.2 shows an example for the stress constraint in which the objective function value is:

$$f(x) \qquad \text{If: } \sigma_{max}(x) \leq \sigma_{crit} \tag{3.1}$$

$$f(x) = f(x)\left(1 + \frac{\sigma_{max}(x) - \sigma_{crit}}{\sigma_{crit}}\right)^3 \qquad \text{If: } \sigma_{max}(x) > \sigma_{crit} \tag{3.2}$$

In which $\sigma_{max}(x)$ is the maximum stress in the design and σ_{crit} is the maximum allowable stress. For the other methods, the design variables were normalized between zero and one. For this method, the design variables are normalized between 10 and 11. The MATLAB function Fminsearch creates the initial simplex by adding 5% of the design variable value for each node, with a minimum of 0.00025. This can create a severely elongated starting simplex, depending on the starting point. Convergence is generally better and faster if the starting simplex is equally spaced. Normalizing between 10 and 11 also results in a much larger starting simplex, which can also be favorable for fast convergence. A few examples for a two dimensional design space are shown in figure 3.1. Note that some of the vertices of the starting simplexes fall outside the bounds. In case the functions are not defined outside the bounds, the starting points should be carefully considered.

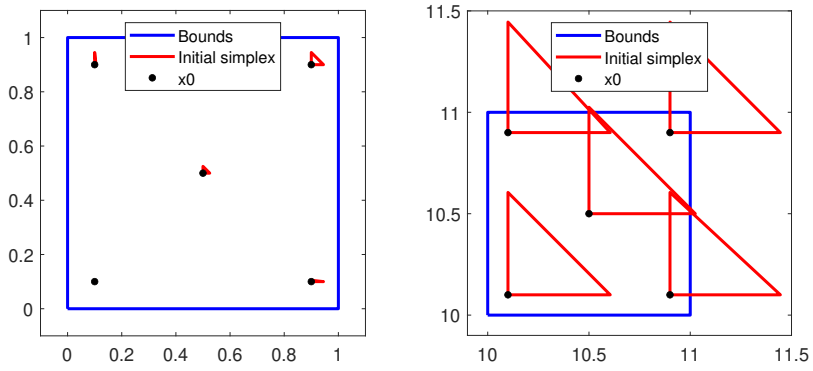


Figure 3.1: Initial simplexes from different starting points for a design space normalized from zero to one on the left and from 10 to 11 on the right.

3.3 Structure of the results

The first part of this research is focused on a simple design with few design parameters, for which all the optimization algorithms introduced in chapter 2 can find the optimum, resulting in a clear comparison of the performance of the different algorithms. In the next part of the research, the complexity of the design problem is increased, mainly by increasing the number of design variables. This leads to a few extra strategies required for successful and reliable optimization. In the last part of this research, the fully developed method is tested on a larger design problem with up to 25 design variables. The results of this research are subdivided into three chapters. The designs and optimization problems are presented at the start of each chapter. The chapters are divided as follows:

- In chapter 4, a simple design with 2-4 design variables is used to analyze the optimization problem extensively and to test the developed methods. The design space is explored graphically. The effect of different choices in the scaling of the objective function, the constraint functions, and the design variables is shown. With the analysis of the finite differences, an appropriate step size for the design variables is selected. The performances of all the algorithms introduced in chapter 2 are compared by graphically showing the convergence rate. From these algorithms, one of the MATLAB fmincon algorithms is selected and used in the follow-up problems, along with the quasi-Newton method.
- In chapter 5, two similar hinges as in chapter 4, are introduced, with eight and ten variables respectively. These models require a different approach to the definition of the stress constraint, where taking the maximum Von Mises stress over the entire hinge is not enough. The second model in this chapter shows a complication where multiple frequency modes can become the lowest frequency within the design space.
- In chapter 6, the capabilities of the developed methods are demonstrated on a reinforced cartwheel hinge design, with up to 25 design variables. In this chapter, some local minima are encountered and analyzed. This chapter also includes graphs that show the performance of the optimization algorithms. The number of function evaluations required for each problem to converge while using the SQP algorithm is shown, resulting in a benchmark for the number of function evaluations as a function of the number of design variables.

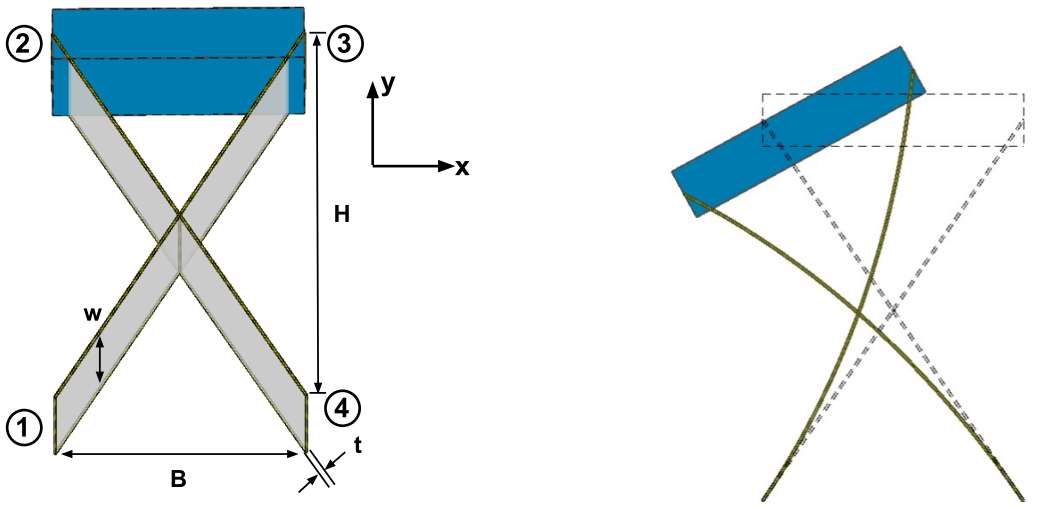
Chapter 4

Cross-hinge

In this chapter, optimization tools are developed, using a simple optimization test case as example. The aim is to show the feasibility and capabilities of the algorithms and methods discussed in the previous chapters. First, the test case model is introduced. Then several analysis steps, that are used to help the optimization process, are discussed and finally, the convergence performance of the algorithms is presented.

4.1 Model

A simple flexure mechanism is defined for an initial study on the optimization of flexures with SPACAR. This cross-hinge has four design variables and will be referred to as the cross-hinge 4p. The simplified cross-hinge is shown in Figure 4.1, and consists of two crossed leaf springs, fixed on one end and attached to a rigid body on the other end. The two leaf springs can move through each other in the model. Although this is physically impossible, the model can be used as a simplified system for the study of the optimization algorithm. A physically feasible variant of such a hinge is usually configured with one leaf spring placed in the middle and in the other direction leaf springs are placed on both sides, to prevent torsional asymmetry.



(a) Cross-hinge including design variables

(b) Cross-hinge at maximum rotation

Figure 4.1: Model of the four parameter cross-hinge. The model of the two parameter cross-hinge is the same but with w and B fixed.

At the third node, a mass and moment of inertia are added, which represents an object attached to this

hinge. The density of the rigid beam connecting the leaf springs at the top is $3000kg/m^3$. The angle of the rotation is indicated as θ. Warping of the leaf springs is set to 'true', for this model and all other models in this work. The material properties of the leaf springs and other model parameters are shown in figure 4.1.

Table 4.1: Model parameters. The mass and moment of inertia terms are applied at node 3.

Parameter	Quantity	Unit
θ_{max}	0.5	rad
ρ	7800	kg/m^3
E	210	GPa
G	70	GPa
m	1	kg
J_{xx}	10^-3	kg m^3
J_{yy}	10^-3	kg m^3
J_{zz}	10^-3	kg m^3

4.1.1 Optimization problem

Since the main drawback of flexure mechanisms is a low stiffness, especially for large stroke hinges, it is interesting to maximize the stiffness in the non-compliant directions, given a prescribed rotation in the compliant direction. For this purpose, using the first parasitic eigenfrequency is ideal, because this takes all stiffness components into account. The model is fixed in the compliant direction, so the first eigenfrequency is parasitic. The optimization problem is formulated as follows:

Find:

$$x = \{H, B, w, t\}$$

Minimize:

$$-f_1(x)$$

Subject to:

$$\begin{aligned} 0.05 \leq H \leq 0.2 \text{ m} \\ 0.05 \leq B \leq 0.2 \text{ m} \\ 0.025 \leq w \leq 0.075 \text{ m} \\ 0.0001 \leq t \leq 0.001 \text{ m} \end{aligned} \tag{4.1}$$

$$\begin{aligned} \sigma_{max}^{\text{VM}}(x) \leq 300 \text{ MPa} \\ M_{max}(x) \leq 2 \text{ Nm} \end{aligned}$$

4.2 Design space and finite difference analysis

4.2.1 Parameter sweep

In figure 4.2, a parameter sweep of the cross-hinge is shown. The first row of plots shows the first natural frequency as output, the second row shows the maximum stress and the third row shows the maximum actuation moment. In each column, a sweep from the lower bound to the upper bound of one of the design variables is shown. Five random nominal settings are used, resulting in five lines per plot. The dashed lines in the stress and moments plots show the constraint limit.

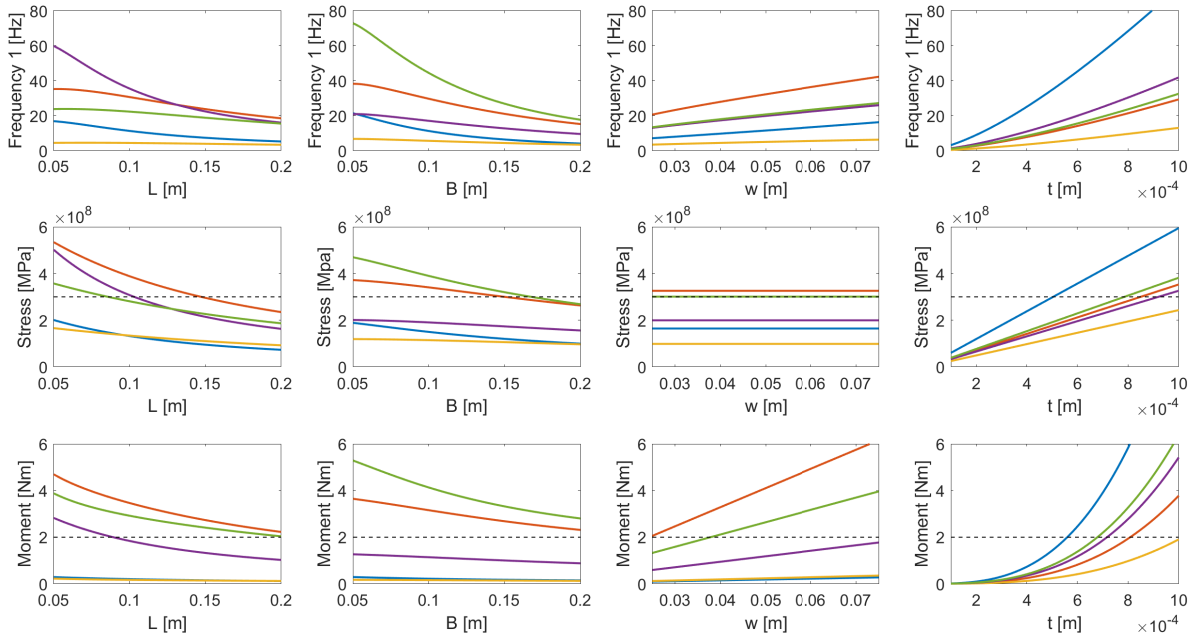


Figure 4.2: Parameter sweep of the objective function and constraints. The constraints are the maximum Von Mises stress and the actuation moment. Dashed lines indicate the constraint limits.

From this sweep, the objective and constraint functions seem continuous over the whole design space. Furthermore, no extreme non-linearities are observed which would call for other definitions or scaling of the parameters. Furthermore, the sweep gives good insight into how the frequency, stress, and moment behave when changing the four parameters. Some of the relations seem to be linear, such as the w as function of frequency, stress, and moment. These relations could potentially be used to simplify the problem.

4.2.2 Visualization using two design variables

A great difficulty of optimization with many design variables is the inability of visualizing the design space. If everything is set-up correctly this is not a problem. However, when looking into a new type of problem, or developing a custom optimization tool, a lack of visualization is a big hindrance. To circumvent this problem, a problem can always be reduced to just two design variables.

For the cross-hinge, the first optimization attempts with all four design variables showed that the width (B) of the cross-hinge and the width (w) of the leaf springs converge to the lower and upper bound respectively. The length (L) and thickness (t) go to some value between the bounds. This makes the latter two more interesting for the reduced problem with two design variables.

A full mapping of the design space of this two parameter cross-hinge is shown in Figure 4.3. The location where the stress and moments reach the constraint limit is also shown in this plot. What stands out is that there is a kink in the frequency map, where the frequency suddenly changes direction. In this location, the frequency is not differentiable. This raises the suspicion that the mode of the first natural frequency changes at this kink. In Section 5.4.1 it is shown for a similar case that this is indeed what happens. In the cross-hinge problem, it happens far from optimum and the direction does not change dramatically, so is not expected to be an issue in this case. However, this phenomenon can cause problems with gradient-based methods.

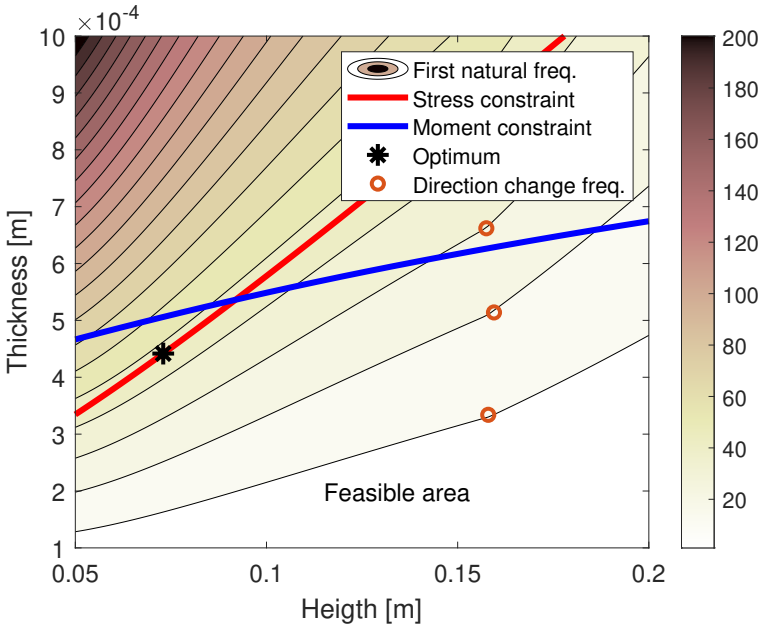


Figure 4.3: Visualization of the design space of the two parameter cross-hinge

Figure 4.3 also shows the optimum found with fmincon. It is noticeable that the stress constraint is close to parallel to the objective function at the location of the optimum. To visualize the local optimum, the objective function value at the stress constraint and moment constraint limits is shown in Figure 4.4. It appears that the earlier found optimum is indeed correct. Note that if the moment constraint was not present, this cross-hinge would have a local minimum at the location where the frequency suddenly changes direction.

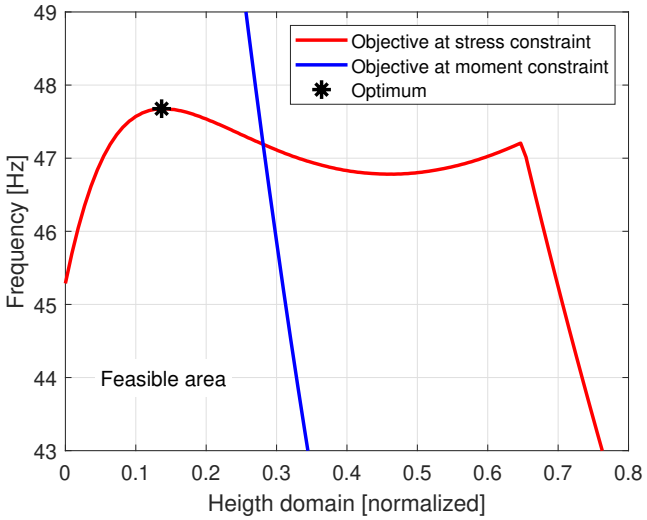


Figure 4.4: The first natural frequency at the stress and moment constraints.

4.2.3 Numerical derivatives

The step size sweep of the objective function with respect to the four design variables is shown in Figure 4.5. This sweep is done in normalized design space, so the step size is the fraction of the domain of the design variable. In these plots, the default finite difference step size used by fmincon is shown by the black dotted line. It can clearly be seen that the forward difference is already somewhat unstable at this

step size. A larger step size should be used. According to these plots, the step size should roughly be taken between 10^{-3} and 10^{-6}, since these values return fairly accurate forward differences. These values are variable dependent, but in this case the range is roughly the same for all four parameters. This range is based on first order derivatives. For second order derivatives, the noise has even more severe effects. Because of this, for calculating the Hessian a step size of 10^{-3} was used.

The results in figure 4.5 were generated after increasing the accuracy of design parameters in the SPACAR input files generated by SPACAR Light. This is necessary in order to get accurate finite differences.

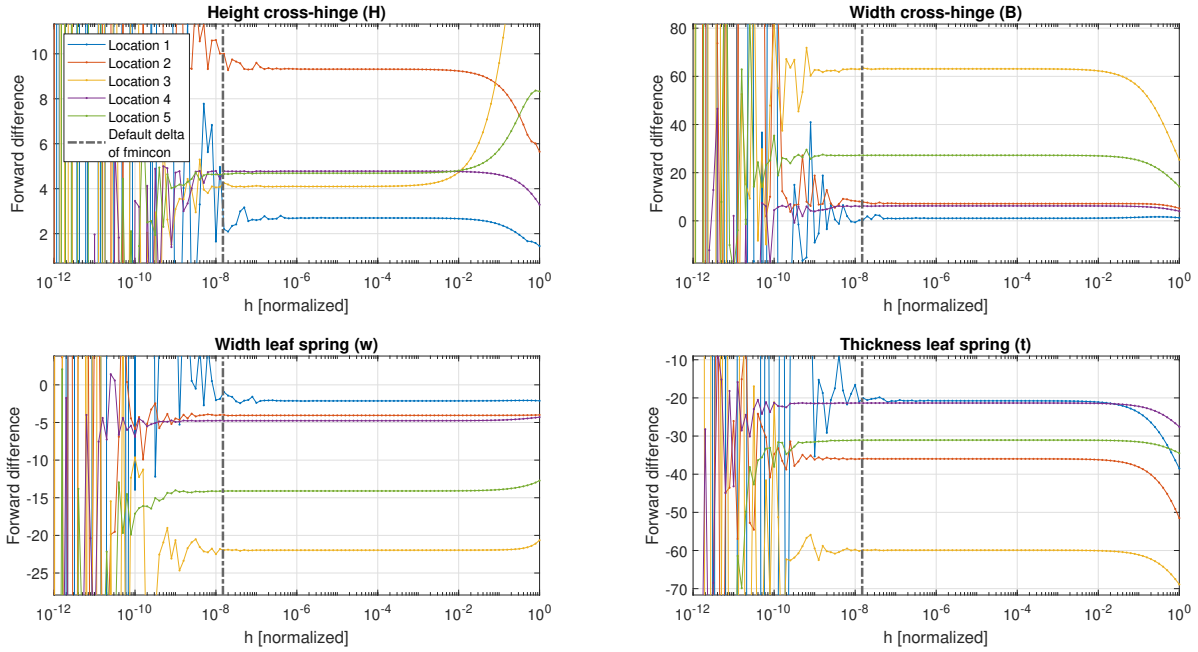


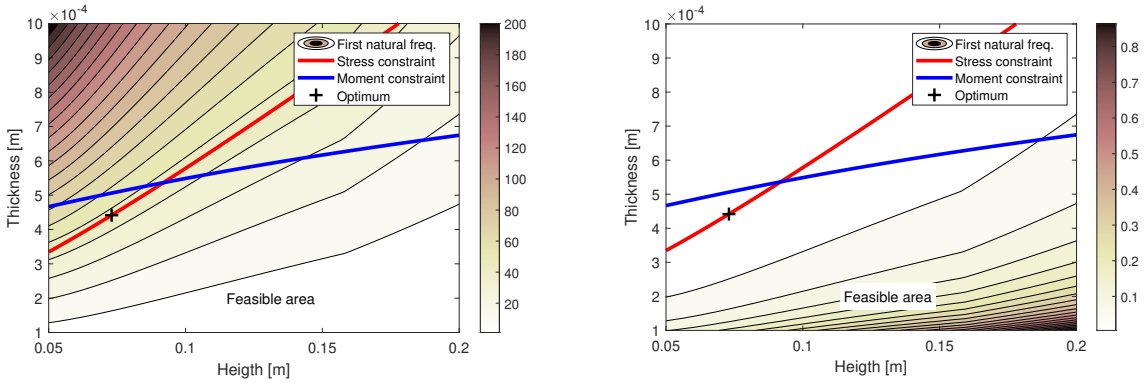
Figure 4.5: Step size sweep of forward differences of the objective function to the four normalized design variables

The gradients of the constraints should also be determined. Similar plots as figure 4.5 were made for the maximum stress and actuation moment and shown in appendix B. The results are similar to the objective function step size sweep, but the acceptable range of step sizes appeared to be larger, so the gradients of the objective function are the limiting factor.

4.3 Optimization results

4.3.1 Influence of scaling and normalization

As briefly mentioned in section 2.1, scaling and normalization of the objective function, design space, and constraints can significantly influence the convergence rate. Optimization generally converges best if the objective function is close to linear, does not span many orders of magnitude, and is close to an order of magnitude one. In previous work [2], the objective function used for optimizing large stroke flexure hinges was the inverse of the first parasitic eigenfrequency. This results in a very non-linear objective domain. An alternative is taking the negative of this frequency instead of the inverse. Both are shown in figure 4.6. The objective function is clearly more evenly spread over the design space for the negative frequency.



(a) Objective function: $-f_1$

(b) Objective function: $\frac{1}{f_1}$

Figure 4.6: Design space comparison between objective functions $-f_1$ and $\frac{1}{f_1}$.

The actual convergence rate of differently scaled objective functions can easily be tested by performing a series of optimization runs from different starting points. For this comparison, fmincon was used with the SQP algorithm. For each different setting, 30 optimization runs were done from different starting points that were generated with the Latin hypercube sampling method. The four parameter cross-hinge was optimized. Figure 4.7 shows the results. The convergence it defined as the evaluation when the difference between the best found solution and the current function evaluation is less than $10^{-3}Hz$. All runs converged well beyond this level of accuracy. This eliminates the influence of the convergence criteria of the algorithm, resulting in a more equal comparison.

Figure 4.7 and A.1 contain box plots. In these plots, the red line gives the median value of the data set. The box spans ±25% from the median. The black lines are called the whiskers and span the whole data set except for outliers, which are shown as red plus signs. Lastly, the boxes are notched. These notches show the 95% confidence range of the median.

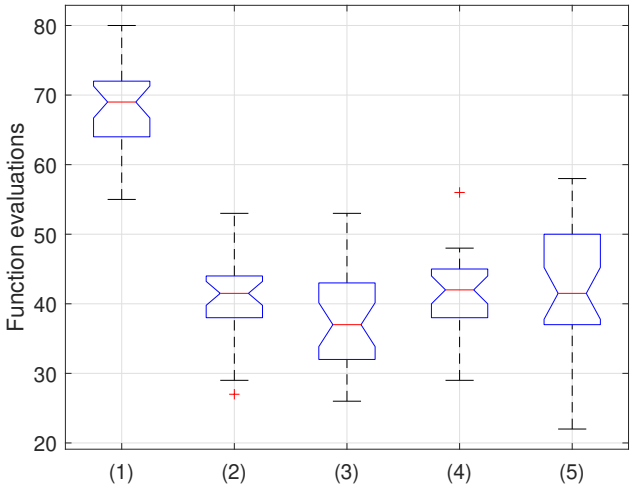


Figure 4.7: Comparison of convergence of differently scaled frequency objective functions. Also contains a box where no normalization has been done on the input variables and on the constraints. The description of the boxes is found in Table 4.2.

Table 4.2: Description of the boxes in figure 4.7

Box nr.	Description
(1)	Objective function: $\frac{1}{f_1}(x)$
(2)	Objective function: $-f_1(x)$
(3)	Objective function: $-f_1(x)/f_1(x0)$
(4)	Same as (3), without normalized design space
(5)	Same as (3), without normalized constraints

It is clear that the objective $-f_1$ (case 2) performs much better compared to the objective $\frac{1}{f_1}$ (case 1). Case (3) gives an even better convergence rate, but only slightly. Not normalizing the design space (case 4) or constraints (case 5), gives a slightly worse performance. For this model, the optimization works fine without normalization. However, in many cases not normalizing the design space and constraint functions can be problematic. In all further optimization results in this research, the design space and constraints are normalized. The objective function $-f_1$ (case 2) for the results in this chapter. Case 3 is used as objective function in the remaining chapters.

Scaling of a compliance objective function

In flexure mechanisms, it can be relevant to optimize the stiffness in specific directions, in which case the compliance C can be minimized. Scaling of this objective function is necessary, because the compliance spans multiple orders of magnitude within the design space. The results of different scalings, with 30 optimization runs each, is shown in Appendix A. In conclusion, it is recommended to scale the compliance by taking the log.

4.3.2 Convergence rate comparison

To compare the performance of each optimization method, the cross-hinge has been optimized 100 times with each algorithm. This was done for the cross-hinge with two and four design variables. The starting points are 100 locations in the design space that were generated with the Latin hypercube method. For each algorithm, the same starting points were used. In figure 4.8, the results are shown for the two variable cross-hinge. The plot shows the median value out of 100 runs, of the difference between the best feasible solution found at a certain iteration, minus the best found solution in all runs, which is therefore

considered to be the global minimum. The median is used instead of the average, because for the logarithmic scale of the convergence measure, outliers dis-proportionally skew the average. To determine if the solution is feasible, it is checked if all the bounds and constraints are satisfied.

The optimization runs are not terminated before the maximum number of function evaluations is reached, such that the convergence criteria do not influence the result. At some runs in fmincon, even though the convergence criteria was set to zero, the run was still terminated before the function evaluation limit was reached. This is because subsequent iterations yielded exactly the same results. It is assumed that even if not terminated, a better solution would not have been found.

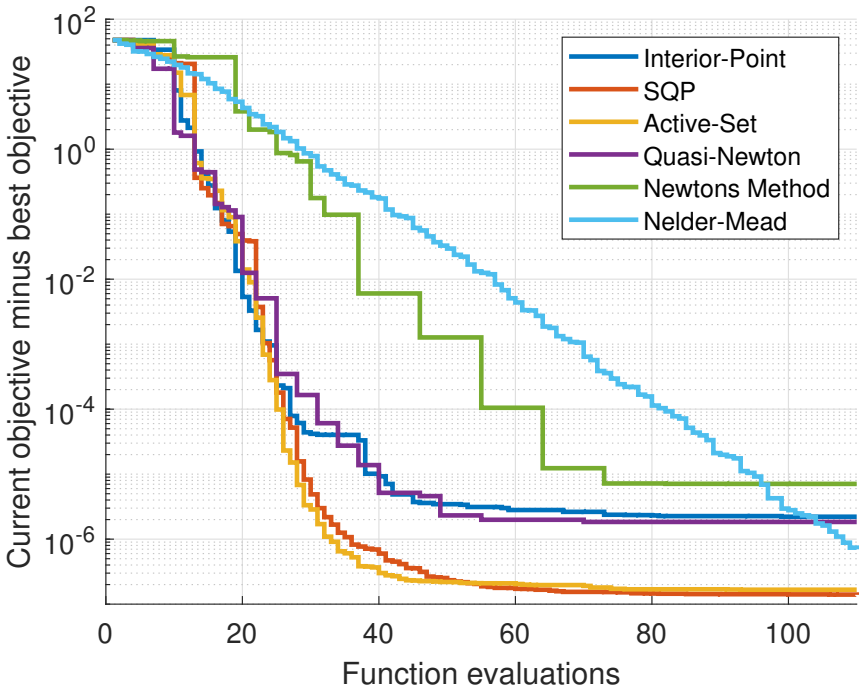


Figure 4.8: Median convergence rate of each optimization algorithm using 100 semi-random starting points. Applied on the two parameter cross-hinge.

For the results, it is important to note that the convergence rate at small differences with the best solution is not of importance. Different algorithms can converge in slightly different ways, resulting in slightly different objective function values. From figure 4.8 it can be concluded that the median convergence rate of all the gradient-based methods using BFGS approximations of the Hessian is the fastest. These methods all perform roughly equal. The Full-Newton method requires more function evaluations, since the Hessian needs to be calculated each iteration. The Nelder-Mead algorithm converges significantly slower compared to the gradient-based methods.

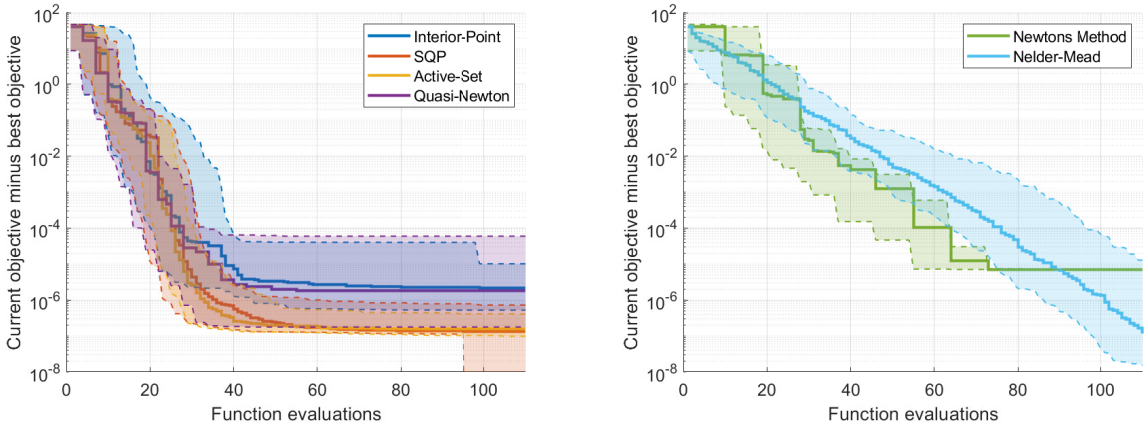


Figure 4.9: Spread of convergence rates of the different algorithms applied on the two parameter cross-hinge. The best and worst 10% of all runs are excluded.

The median performances of the fmincon algorithms and quasi-Newton method are approximately equal. Consistency is another measure to differentiate between algorithms. In figure 4.9, the spread of the convergence of different attempts is shown. The dashed lines give the best and worst converging attempts. The SQP is the most consistent fmincon algorithm and is the preferred fmincon algorithm based on this information. It is also noteworthy that each Newton's method run ends in exactly the same optimum. Furthermore, the Nelder-Mead has the largest spread as expected and even the best attempt converges slower than the worst quasi-Newton and fmincon attempt.

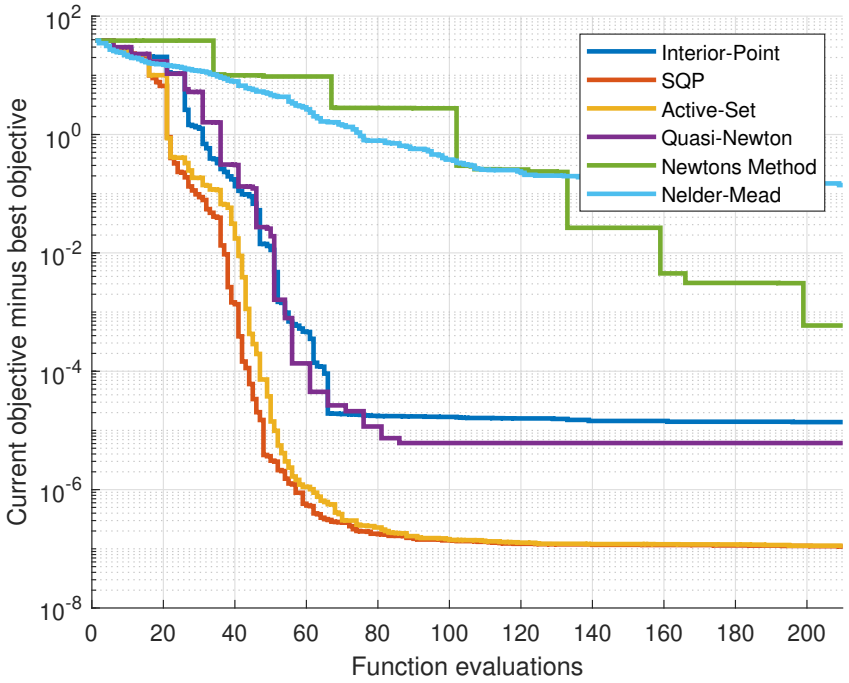


Figure 4.10: Median convergence rate of each optimization algorithm using 100 semi-random starting points. Applied on the four parameter cross-hinge.

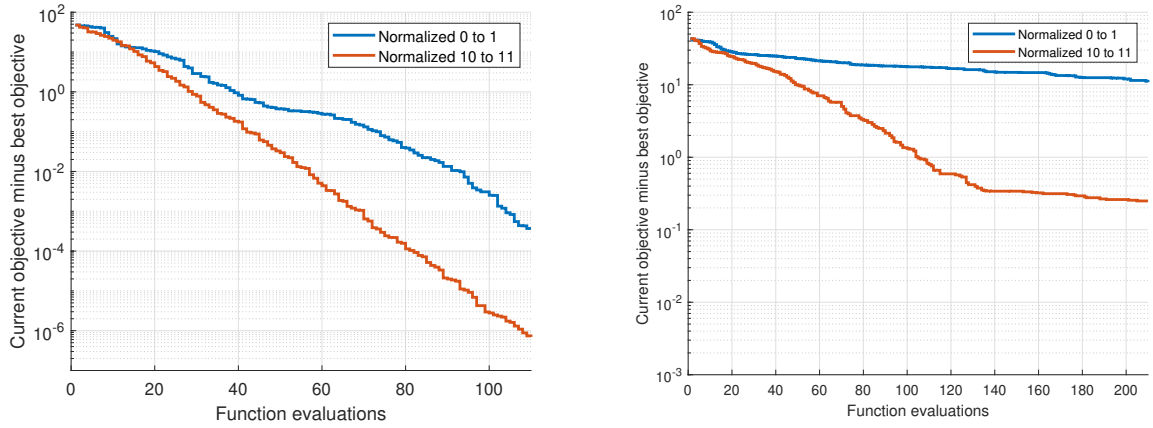
The same convergence plot was also made for the four parameter cross-hinge, as shown in figure 4.10. In this case, the different BFGS based algorithms diverge slightly in convergence rate. The SQP and

active-set converge the fastest and nearly at the same rate. This is not surprising since these algorithms are nearly identical. SQP is the most refined algorithm of the two and performs slightly better. The quasi-Newton method converges significantly slower compared to the SQP algorithm. There are multiple possible reasons why the SQP algorithm converges faster. The trust-region fmincon uses is adjusted during the optimization run, whereas the trust region in the quasi-Newton method is constant. Fmincon also sometimes performs a line-search, which is not implemented in the quasi-Newton method.

4.3.3 Nelder-Mead convergence

In section 2.2, the convergence of optimization of a cross hinge using Nelder-Mead in previous research was discussed. Using the Nelder-Mead algorithm with the Matlab function fminsearch, a similar optimization was done on the cross hinge with four design variables, for a total of 100 runs. After 135 function evaluations, 49% of the runs were converged within 1% of the global optimum and 67% converged within 5% of the global optimum. This is an improvement over the algorithm used by Naves [4]. In conclusion, this implementation of Nelder-Mead is at least representative for the algorithms used in previous work.

Figure 4.11 illustrates the difference in the performance of Nelder-Mead between a normalized design space from 0-1 and from 10-11, as discussed in section 3.2.3.



(a) Two design variables

(b) Four design variables

Figure 4.11: Median convergence rate out of 100 runs of the Nelder-Mead algorithm. The performance between the two different normalization ranges is compared.

Chapter 5

Complications in cross-hinge variants

In the previous chapter, the optimization of a simple cross hinge with 4 design variables at most was studied. That was used to study how to numerically determine the derivatives, and to get an impression of the effect of optimization algorithm and objective function definition on the convergence rate. By increasing the complexity of the problem, other issues may become apparent. By introducing more design variables, the variety of solutions throughout the design space will also increase. This causes certain effects when using frequencies and stresses in the constraint and/or objective function. These effects will be shown in this chapter by increasing the number of design variables in the cross-hinge problem.

5.1 Models

Two different models are analyzed in this chapter. Both are introduced in this section.

5.1.1 The eight parameter cross-hinge model

In the previous chapter, the cross-hinge design optimization used four design variables. With some model modifications, the number of design variables is increased to eight. A schematic overview of this modified cross-hinge is shown in figure 5.1. Both leaf springs can have their own thickness and width. The height and width of the cross hinge can also be varied on all sides, with respect to the center. Note that this extra design freedom can result in asymmetry in the yz and xz-plane. Since the model is asymmetric, due to the mass and inertia at node 3, it is expected that the optimal geometry will be asymmetric as well. The advantages of using the same simple cross hinge as the base design, are relatively fast function evaluations and predictable behavior.

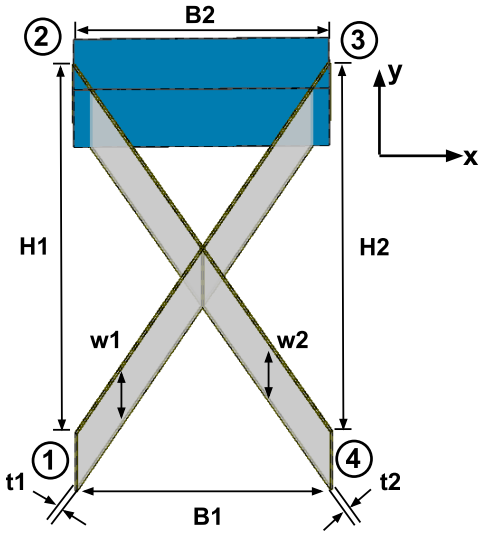


Figure 5.1: Model of the eight parameter cross-hinge.

This model is identical to the cross-hinge introduced in section 4.1, apart from the extra design variables. For the model parameters, see Table 4.1.

5.1.2 Multi-beam cross-hinge model

To enable an increasing number of design variables on a relatively simple design, both leaf springs of the cross-hinge are divided into n beams of equal length. The load case and constraints are the same as in the previous cross-hinge designs. For simplicity, only the thickness of each beam element is used as a design variable. All other variables are fixed. The height H and width B of the cross hinge are both 100 mm and the width w of all beams is 50 mm. The thicknesses can range from 0.2 mm to 2 mm. An example of this cross hinge with $n = 5$ is shown in figure 5.2.

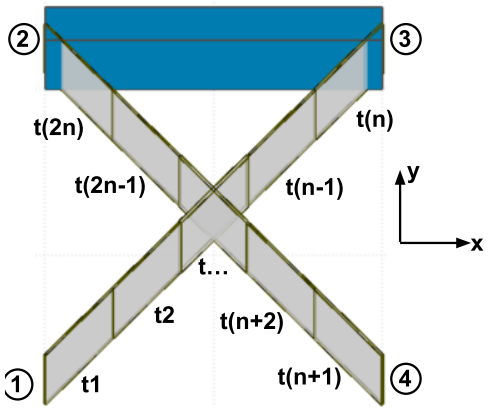


Figure 5.2: Model of the multi-beam cross-hinge with n beams of variable thickness

5.2 Optimization problems

Eight parameter cross-hinge

The optimization problem for the eight parameter cross-hinge is defined as:

Find:

$$x = \{H1, H2, B1, B2, w1, w2, t1, t2\}$$

Minimize:

$$-f_1(x)/f_x(x_0)$$

Subject to:

$$\begin{aligned} 50 \leq (H1, H2) \leq 200 \text{ mm} \\ 50 \leq (B1, B2) \leq 200 \text{ mm} \\ 25 \leq (w1, w2) \leq 75 \text{ mm} \\ 0.1 \leq (t1, t2) \leq 1 \text{ mm} \end{aligned}$$

$$\begin{aligned} \sigma_{max}^{VM}(x) \leq 300 \text{ MPa} \\ M_{max}(x) \leq 2 \text{ Nm} \end{aligned}$$

Multi-beam cross-hinge

The optimization problem for the multi-beam cross-hinge is defined as:

Find:

$$x = \{t1, ..., t(2n)\}$$

Minimize:

$$-f_1(x)/f_x(x_0)$$

Subject to:

$$0.2 \leq (t1, ..., t(2n)) \leq 2 \text{ mm}$$

$$\begin{aligned} \sigma_{max}^{V}(x) \leq 300 \text{ MPa} \\ M_{max}(x) \leq 2 \text{ Nm} \end{aligned}$$

5.3 Results of the eight parameter cross-hinge

5.3.1 Separate stress constraints

The first optimization attempts of this design lacked convergence. Multiple runs from different starting points did not result in a consistent optimum. In order to find the root cause of this issue, a parameter sweep was performed. The issue is already clear with just one sweep from the starting point:

$$x0 = \{0.1 \text{ m}, 0.1 \text{ m}, 0.1 \text{ m}, 0.1 \text{ m}, 0.05 \text{ m}, 0.05 \text{ m}, 0.0005 \text{ m}, 0.0005 \text{ m}\}$$

Parameter sweeps from different starting points yielded similar results. While the objective function and moment constraint did not show any surprising behavior, the stress constraint sweeps did. The red lines in Figure 5.3, represent the maximum Von Mises stress as function of the design variables. This maximum stress was used for the stress constraint in the previous chapter. The maximum stress is clearly not continuously differentiable in this design space. Since a requirement for gradient-based optimization is a sufficiently smooth function, this is likely to be the main problem.

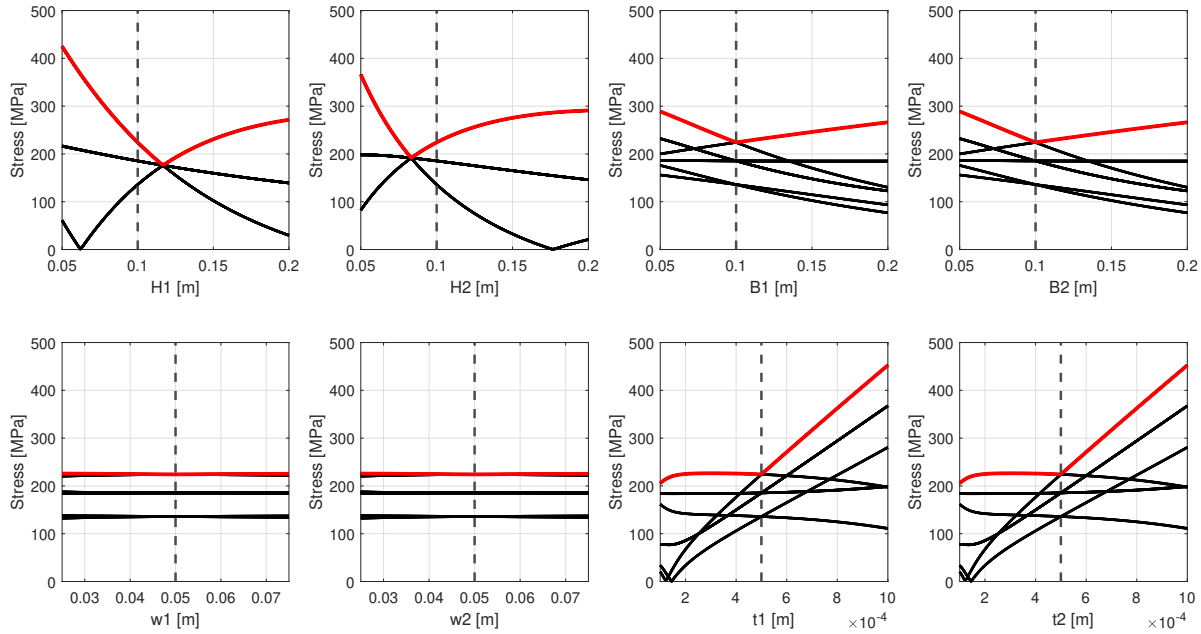


Figure 5.3: Parameter sweep of the stress constraints for the eight parameter cross hinge. The Von Mises stress at 32 locations is shown, most of which overlap, due to symmetry in the model. The maximum Von Mises stress is shown in red. The dashed lines show x_0.

The Von Mises stresses at the vertices of the beams, that are not the maximum stress, are shown in black. There are a total of 4 flexible beams in this cross hinge, so there are 32 stress constraints. Many of these stresses overlap, partly due to the symmetry of the starting geometry. What happens is that the location of the maximum stress changes within the design space. The solution for this problem is to treat the Von Mises stress at all locations as separate constraints. However, the stress is calculated by SPACAR at many points in each beam. To keep the number of constraints limited, only the Von Mises stress at the eight vertices of each beam is used as a separate constraint. This has proved to be sufficient for optimizing most problems, since the maximum Von Mises stress is generally located at one of these vertices. It might be possible that for some designs the maximum stress will not appear at the vertices. In this case, the choice of constraints should be reconsidered.

As can be seen, the separate stresses are smooth. However, they do change direction at zero since the stress at this point changes from tensile stress to compressive stress or vice versa. The exact influence of this phenomenon was not researched, but it did not cause serious issues in the optimizations that were done.

5.3.2 Convergence results

The median convergence of the eight parameter cross-hinge is shown in figure 5.4. Each algorithm is run from 30 different starting points, which were generated with the Latin hypercube method. Since the SQP algorithm performed best in the previous chapter and the quasi-Newton algorithm has its own merits, as will be discussed in section 5.4.1, only these two algorithms are tested on this model, and the models presented in chapter 6. In this plot, and the convergence plots in section 6.2.2, function evaluations that violate the stress constraint by less than 0.1 MPa are allowed. However, the best solution to which all evaluations are compared is strictly feasible. By taking the absolute value of the difference between the current objective value and the best objective value, the slightly infeasible solution cannot cause negative values.

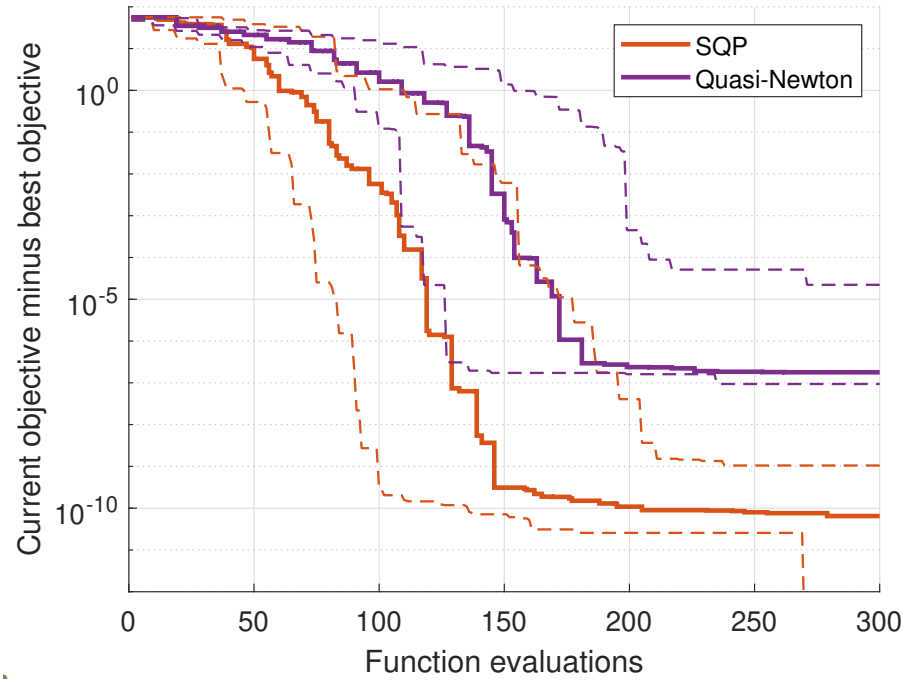


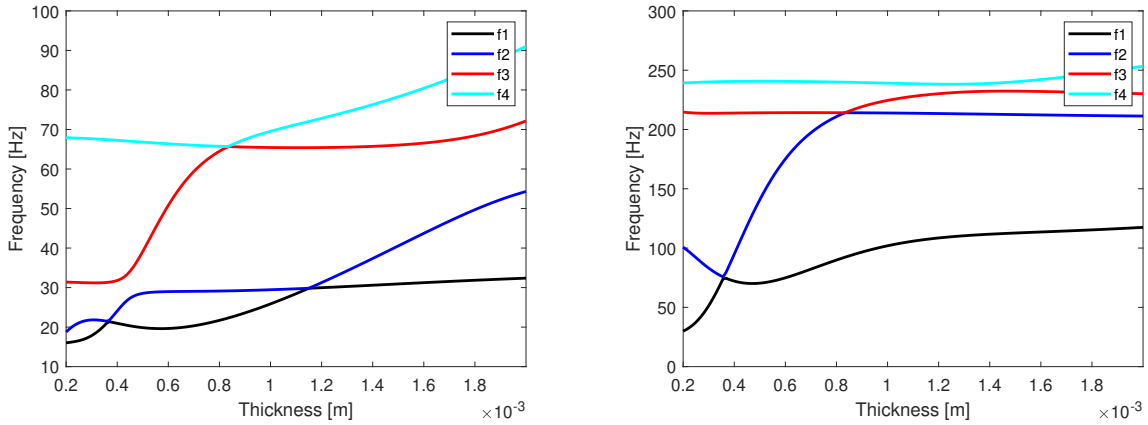
Figure 5.4: Median convergence out of 30 runs of the eight parameter cross-hinge. Dashed lines indicate the range excluding best and worst 10% of all runs.

5.4 Results of the multi-beam cross-hinge

In this section, a problem with multiple frequency modes, which alternately are the lowest frequency within the design space is discussed. This is a particularly major problem within this specific model, although it might be encountered in other models as well. In Chapter 6, models with more design variables are optimized where this issue is only encountered sporadically.

5.4.1 Separating frequency modes

Initial optimization tests with $n >= 2$ using the SQP algorithm, showed inconsistent convergence, and different runs that did not converge to a local optimum. The likely problem is revealed by displaying the first four eigenfrequencies in a parameter sweep over the thickness.



(a) Sweep of one of the thicknesses for the $n = 2$ model. (b) Sweep of one of the thicknesses for the $n = 5$ model.

Figure 5.5: Parameter sweep of the natural frequencies at a random parameter configuration for the multi-beam cross-hinge with 2 beams per leaf spring and the model with 5 beams per leaf spring

In figure 5.5, a parameter sweep of the first four eigenfrequencies is shown. For the left sub-figure the $n = 2$ model is used, so with a total of four beams, and for the right sub-figure the $n = 5$ model is used, with a total of ten beams. For both sub-figures, only the thickness of one of the beams is varied. Sweeps of the thickness of the remaining beams yielded comparable results. It is clear that the lowest eigenfrequency, used in the objective function, is not continuously differentiable. Figure 5.5 shows that this is caused by different frequency modes which take over when increasing or decreasing the dimension of a design variable. This can be a problem for reliable gradient-based optimization.

A solution for this problem could be to separate frequency modes. Analogous to the separate stress constraints, the frequency modes can be calculated separately. This is not applicable within fmincon, however, it is possible to implement this in the Quasi-Newton method.

There are possible methods to keep track of different frequencies in SPACAR, but it is not straightforward to implement this in a reliable manner. It was decided that this falls outside the scope of this research. But it is recommended to explore this option for when the eigenfrequencies are used as constraints or objective function.

Chapter 6

Cartwheel

In this chapter, the developed optimization methods are used to optimize a useful realistic design, with a large number of design variables. The designs to be optimized in this chapter are all variations of the cartwheel hinge, which is a hinge with two leaf springs, oriented at an angle and connected at the center. The pivot shift of the cartwheel hinge during a stroke is very small compared to the cross hinge with disconnected springs [15]. This can be a major advantage of this type of hinge. A simple cartwheel hinge, with only four design variables (H B t and a rotation applied on the moment of inertia tensor), was optimized by Wiersma [2]. One of the aims of this chapter is to show the benefit of increasing the design freedom when it comes to the performance of the hinge.

In the previous chapter, it was shown that using the frequency as objective function can lead to optimization problems if multiple frequency modes are the lowest within the design space. The eigenfrequency is still used as objective function in this chapter, since it still is a very useful performance objective. For all models presented in this chapter, the presumably global optimum could be found.

Adding reinforcements to leaf springs can increase the support stiffness and the first parasitic eigenfrequency. This has been studied and applied on parallel leaf spring configurations [16]. The application of reinforcements might also benefit the cartwheel hinge. In section 6.1, the basic cartwheel is compared to a reinforced cartwheel, to show the improvement performance. The reinforced cartwheel is optimized with several levels of design freedom, increasing the number of design variables which increases the design performance. For all the models in the previous chapters, the rotation on the hinge was only modeled in the positive counterclockwise direction. This is a useful simplification, but in section 6.1.4, it is shown that for geometries that are not symmetrical in the y-z plane, this simplification should be avoided.

6.1 Models

Figure 6.1 schematically shows the two cartwheel concepts. The width (w) is defined in z-direction, and is therefore not shown in this graph. For some of the models, individual 'arms' of the hinge, have their own design variables. The number of the design variable corresponds with the node of the respective arm. In section 6.1.1, a three parameter cartwheel and six parameter reinforced cartwheel are both optimized and compared. In section 6.1.2, first the symmetry over the xz-plane is removed and the design variable w is added, resulting in the 13 parameter reinforced cartwheel. Also removing the symmetry over the yz-plane resulted in the 25 parameter cartwheel

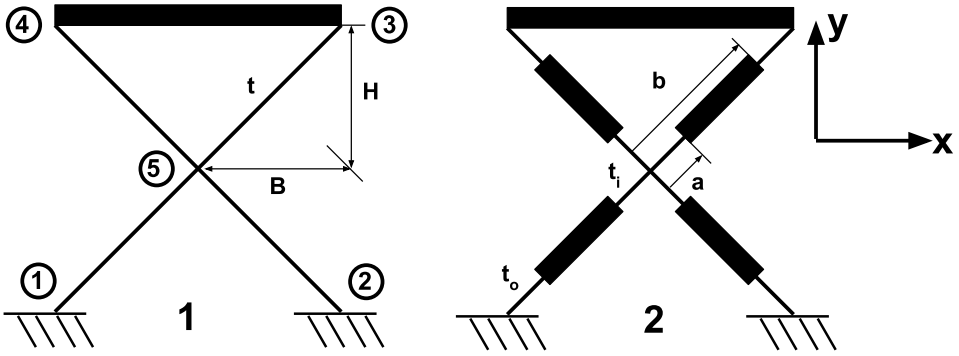


Figure 6.1: Schematic drawing of the regular and reinforced cartwheel concepts

The model parameters of the cartwheel hinges used in this chapter are presented in Table 6.1. The moment of inertia and mass applied at the pivot of the hinge used on this cartwheel was also used in the work of Wiersma [2] and Naves [4]. This mass and inertia represent an L-shaped robot arm rotating around the pivot point of the hinge. The density for the rigid beams and leaf springs is the same. The thickness of the reinforcements is 2.5 mm. The θ_{max} of $\pm 10°$, was only applied for the full range of motion 25 parameter cartwheel, starting in section 6.1.4. For all other cartwheel hinges θ_{max} is only modeled in the positive direction. For the three variable cartwheel and six variable reinforced cartwheel, the width (w) of the hinge was fixed at 75 mm.

Table 6.1: Model parameters. The mass and moment of inertia terms are applied at the pivot of the hinge, rigidly attached to node 4.

Parameter	Quantity	Unit
θ_{max}	± 10	°
ρ	7800	kg/m^3
E	210	GPa
G	70	GPa
m	0.574	kg
J_{xx}	$3.760^{-}3$	kg m^3
J_{yy}	$3.528^{-}2$	kg m^3
J_{zz}	$3.826^{-}3$	kg m^3

6.1.1 Simple and reinforced cartwheel comparison

Optimization problems

The optimization problem first model introduced in the previous section, the simple cartwheel, is formulated as:

Find:

$$x = \{t, B, H\}$$

Minimize:

$$-f_1(x)/f_x(x_0)$$

Subject to:

$$0.3 \leq t \leq 2 \text{ mm}$$
$$20 \leq B \leq 50 \text{ mm}$$
$$20 \leq H \leq 50 \text{ mm}$$

$$\sigma_{max}^{VM}(x) \leq 300 \text{ MPa}$$

The optimization problem for the reinforced cartwheel is defined as:

$$x = \{t_i, t_o, a, b, B, H\}$$

Minimize:

$$-f_1(x)/f_x(x_0)$$

Subject to:

$$0.3 \leq t_i \leq 2 \text{ mm}$$
$$0.3 \leq t_o \leq 2 \text{ mm}$$
$$0.1 \leq a \leq 0.5$$
$$0.6 \leq b \leq 0.95$$
$$20 \leq B \leq 50 \text{ mm}$$
$$20 \leq H \leq 50 \text{ mm}$$

$$\sigma_i^{\text{VM}}(x) \leq 300 \text{ MPa} \quad \forall i = 1..N_\sigma$$

Results

Note that a and b are defined as ratios of the length of the arm. The optimized models are shown in Figure 6.2. Both were optimized multiple times from different starting points, resulting in a consistent optimum. The reinforcements appear to increase the performance of the hinge. Therefore, the reinforced design will be further optimized in the next sections, with more design variables.

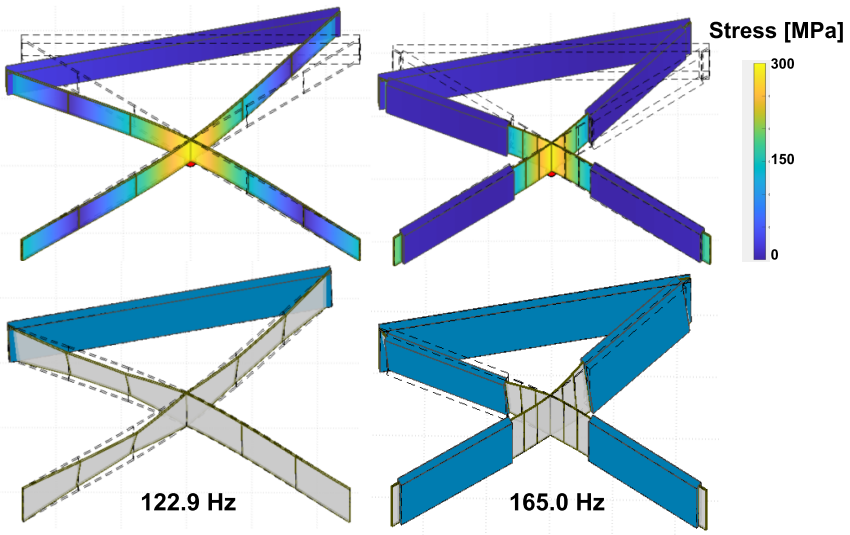


Figure 6.2: Optimized designs of the three parameter cartwheel and six parameter reinforced cartwheel hinges. The top row shows the Von Mises stress distribution and the bottom row the first frequency mode, both at the maximum rotation.

6.1.2 Increased number of design variables

6.1.3 Optimization problems

More design variables can be introduced for the reinforced cartwheel, to allow greater design freedom. The first logical design is to give the top and bottom arms individual design variables while maintaining symmetry over the yz-plane. The width in the z-direction (w) of the entire hinge is also added as design variable. This results in a total of 13 design variables. The corresponding optimization problem is defined as:

Find:

$$x = \{t_{i1}, t_{i4}, t_{o1}, t_{o4}, a_1, a_4, b_1, b_4, B_1, B_4, H_1, H_4, w\}$$

Minimize:

$$-f_1(x)/f_x(x_0)$$

Subject to:

$$\begin{gathered} 0.3 \leq (t_{i1}, t_{i4}) \leq 2 \text{ mm} \\ 0.3 \leq (t_{o1}, t_{o4}) \leq 2 \text{ mm} \\ 0.1 \leq (a_1, a_4) \leq 0.5 \\ 0.6 \leq (b_1, b_4) \leq 0.95 \\ 20 \leq (B_1, B_4) \leq 50 \text{ mm} \\ 20 \leq (H_1, H_4) \leq 50 \text{ mm} \end{gathered}$$

$$\sigma_i^{\text{VM}}(x) \leq 300 \text{ MPa} \quad \forall i = 1..N_\sigma$$

For the next cartwheel hinge, all four arms of the hinge have individual design parameters, and the w is also used as design variable. This hinge contains 25 design variables. The objective function and constraints are comparable to the previous design, but the design variable vector is now:

$$x = \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{o1}, t_{o2}, t_{o3}, t_{o4}, a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, B_1, B_4, H_1, H_2, H_3, H_4, w\}$$

Results

Figure 6.3, shows the results of the three cartwheels with an increasing number of design variables. Each optimization was run 15 times from different starting locations. The six parameter cartwheel always converged to the presumed global optimum, the thirteen parameter cartwheel converged 13 times to the presumed global within the function evaluation limit of 600 and the 25 parameter cartwheel converged nine times to the presumed global optimum within the function evaluation limit of 1500.

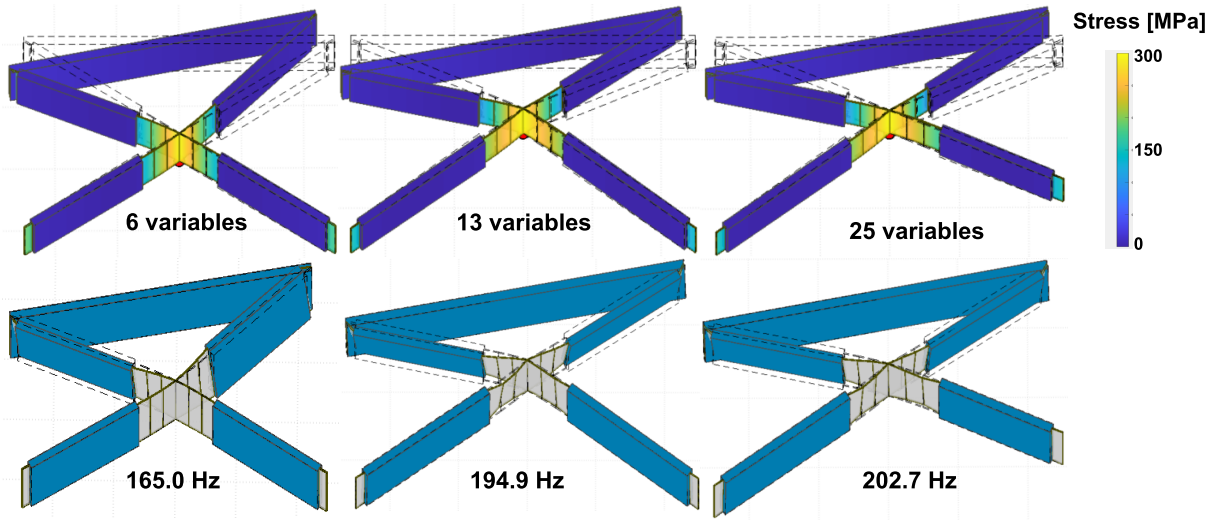


Figure 6.3: Comparison of performance of 6, 13, and 25 parameter reinforced cartwheel. The top row shows the Von Mises stress distribution and the bottom row the first frequency mode, both at the maximum rotation.

The 13 parameter cartwheel improved performance with 29.9 Hz over the six parameter cartwheel hinge. The 25 parameter increases performance by a further 7.8 Hz. This shows that increasing the number of design variables indeed benefits the design. With increased design freedom, the performance should always increase or at least match the performance of the model with fewer design variables. If this is not the case, the optimization algorithm is converged to a local optimum.

6.1.4 Symmetrical rotation

Flexure hinges are generally designed to enable the same maximum rotation to both sides of the resting position. Until this point, the optimizations were performed with only a one-sided positive rotation, because this simplifies the model. For symmetrical designs with a symmetrical load, the result will be representative for the rotation to both sides. However, when the design is allowed to become asymmetrical, it will perform better in the direction where the rotation was modeled and sacrifices performance in the other direction. This is demonstrated in figure 6.4, which shows the frequency over the range of the rotation, for both cases.

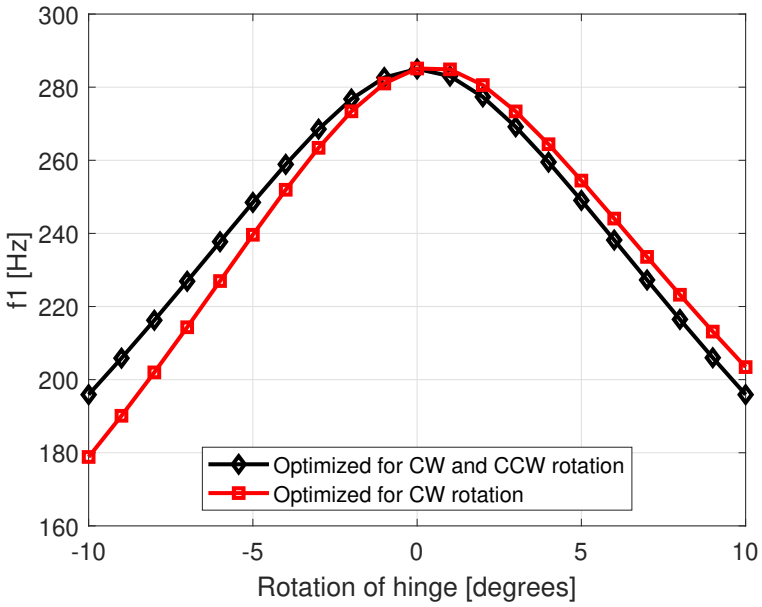


Figure 6.4: Comparison of the first eigenfrequency at all rotation angles for the optimized geometry of the 25 parameter reinforced cartwheel, where one sided or two sided rotation was modeled.

Modeling the rotation in both directions makes the optimization process slightly more complicated. The minimum natural frequency over the whole range of motion can still be used as the objective function. However, for a one-sided rotation, the minimum natural frequency is always found at the maximum rotation angle. While with a two-sided rotation, the maximum can be found at either the positive or negative rotation. This behavior can introduce kinks in the objective function, where the objective is not continuously differentiable. The proposed solution is to define the objective as the natural frequency in one of the maximum deflected positions and to constrain the frequency in the other deflected position to be equal or higher. In that case, it is possible that a better design is found when the location of the objective function and constraint is reversed. To make sure the best solution is found, both cases should be optimized.

The first eigenfrequency for the optimized hinge taking both rotations into account, is 195.9 Hz at both maximum deflections. The first eigenfrequency of the other design is 203.4 Hz at the maximum positive rotation, but only 178.9 Hz at the maximum negative rotation. The optimal design found while taking both rotations into account, is nearly identical to the optimized design found for the thirteen parameter cartwheel, taking only the positive rotation into account as shown in figure 6.3. The difference in performance is small and partly due to numerical discrepancies from SPACAR, by modeling different the different rotations. Adding the extra design freedom does not add any significant performance, due to the near symmetrical nature of the model and design space, in the y-z plane.

6.1.5 Asymmetrical design space

It was shown that full design freedom with 25 design parameters is excessive for the symmetrical design space with a symmetrical problem definition that accounts for the rotation in both directions. However, if the design problem would be more complicated, for instance by introducing an asymmetrical design space, the extra design variables become relevant. Both symmetrical and asymmetrical design spaces are shown in Figure 6.5.

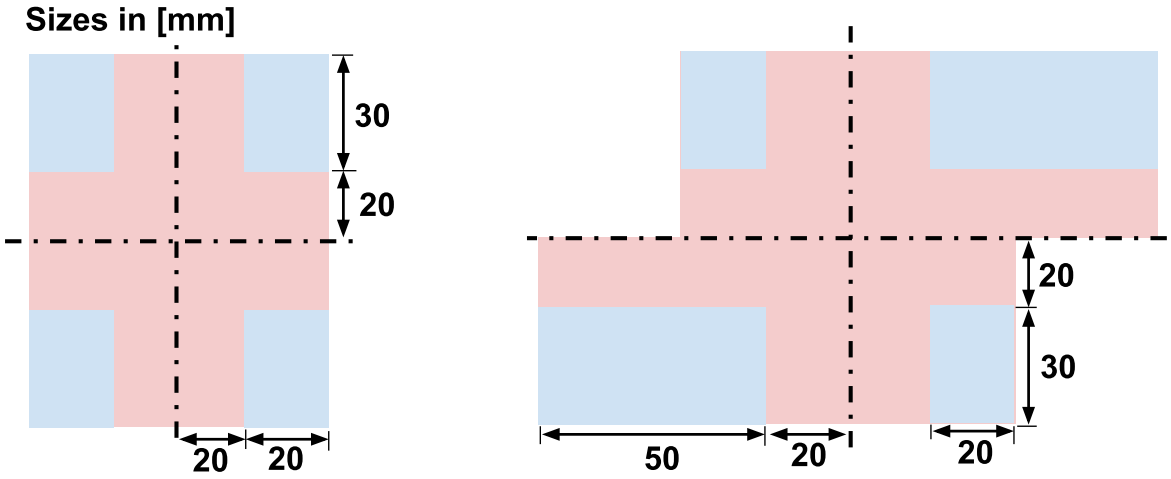


Figure 6.5: Symmetrical and asymmetrical design spaces, in which the blue areas represent the possible locations of the end of the leaf springs.

In this case, two arms of the cartwheel can extend up to 70mm from the center in x-direction, while the other two can only extend up to 40mm. The optimization problem for the model with asymmetrical design space is defined as:

Find:

$$x = \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{o1}, t_{o2}, t_{o3}, t_{o4}, a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, B_1, B_4, H_1, H_2, H_3, H_4, w\}$$

Minimize:

$$-f_1(x)/f_x(x_0), \quad \text{at: } -\theta_{max}$$

Subject to:

$$\begin{aligned}
0.3 \leq (t_{i1}, t_{i2}, t_{i3}, t_{i4}) &\leq 2 \text{ mm} \\
0.3 \leq (t_{o1}, t_{o2}, t_{o3}, t_{o4}) &\leq 2 \text{ mm} \\
0.1 \leq (a_1, a_2, a_3, a_4) &\leq 0.5 \\
0.6 \leq (b_1, b_2, b_3, b_4) &\leq 0.95 \\
20 \leq B_1 &\leq 70 \text{ mm} \\
20 \leq B_2 &\leq 40 \text{ mm} \\
20 \leq B_3 &\leq 70 \text{ mm} \\
20 \leq B_4 &\leq 40 \text{ mm} \\
20 \leq (H_1, H_2, H_3, H_4) &\leq 50 \text{ mm}
\end{aligned}$$

$$\begin{aligned}
\sigma_i^{\text{VM}}(x) &\leq 300 \text{ MPa} \quad \forall i = 1..N_\sigma \\
f_1 \quad \text{at: } -\theta_{max} &\leq f_1 \quad \text{at: } \theta_{max}
\end{aligned}$$

In figure 6.6 a comparison of optimized cartwheels is shown, where the left is symmetrical, and on the right, the extra design space is utilized. The extra design freedom gives a significant performance boost, with an increase of 32.5 Hz.

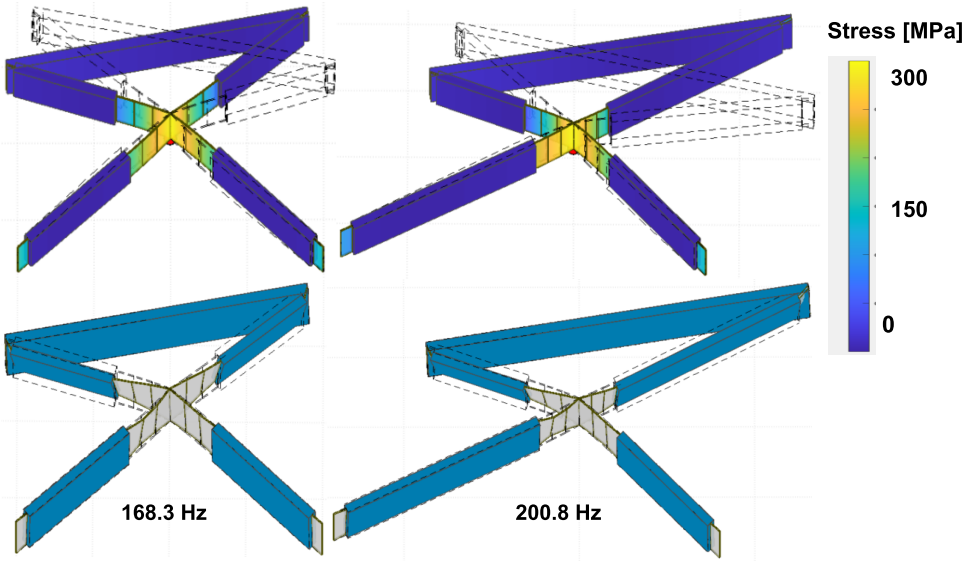


Figure 6.6: Comparison of the optimized symmetrical reinforced cartwheel on the left and the reinforced cartwheel with a modified design space on the right. Taking the rotation in positive and negative directions into account. The top row shows the Von Mises stress distribution and the bottom row the first frequency mode, both at the maximum rotation.

This model of the cartwheel hinge, with the asymmetrical design space, was used in the next section, for generating the convergence results of the 25 design variable problem.

6.2 Results

In this section, the convergence performance of the optimization of the 13 and 25 variable designs is presented. These optimizations were run from 15 starting points. The optimization is done with the SQP algorithm.

6.2.1 Starting points

If the only minimum within the design space, is the global minimum, all optimization runs should converge to the same point for every combination of starting parameters within the design space. It is shown in the next section that this is not the case for the 25 parameter reinforced cartwheel. In this design, there are groups of similar design variables. An example of such similar design variables are the thicknesses of the four inner leaf springs. It is hypothesized that for this design, starting from a symmetrical point, where similar variables have the same value, will result more often in convergence to the global optimum, even though the optimum will not be fully symmetrical.

6.2.2 Convergence results

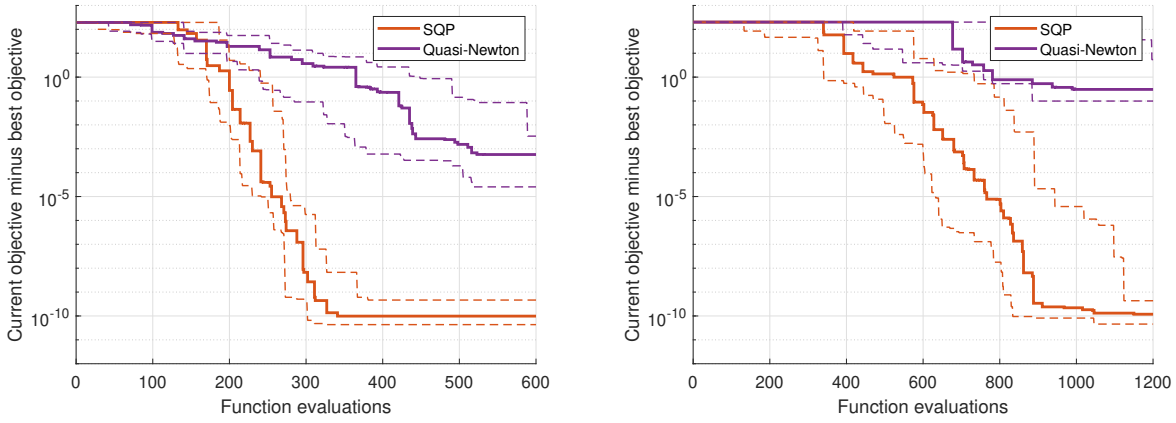
The cartwheel has been successfully optimized with 13 and 25 parameters. The 13 parameter with 15 starting points defined with the Latin hypercube method, was capped at 600 function evaluations. The first series of optimizations was done with the starting points generated with the Latin hypercube (LHC) method. The second series was done with symmetrical starting points. The optimization runs of the 25 parameter asymmetrical design were capped at 1500 function evaluations. Three different series of optimizations with 15 runs each, were done on this model. In the first series, the eigenfrequency during the full stroke was taken as objective function. In the second series, the eigenfrequency at the maximum positive rotation was used as objective function, while the eigenfrequency at the maximum negative rotation was constrained. In the third series, the same settings as in the second series are used, but the starting points are symmetrical. The results are shown in table 6.2. The best solution found is presumed to be the global optimum. If the norm of the resulting set of design variables of a run minus the set

of design variables of the best solution is smaller than 10^{-2}, in normalized space, the global optimum is considered to be found. Convergence to a local optimum is presumed when the search is terminated before reaching the function evaluation limit, but the global optimum is not found. If the maximum number of function evaluations is reached, but the norm of the resulting set of design variables minus the resulting set of design variables of another local minimum is smaller than 10^{-2}, in normalized space, a local optimum is considered to be found. For the termination criteria, the default settings of fmincon were used. The presumed global optimum and several presumed local optima are further analyzed in the next section, in which the optimality conditions are also checked.

Table 6.2: Results of the different optimization series, discussed in this section, using the SQP algorithm.

Series of 15 runs	Global optimum found	Presumed local optimum found	Not converged in fun. eval. limit
13p, LHC start	13	-	2
13p, sym. start	15	-	-
25p f_1 full stroke, LHC start	6	7	2
25p f_1 constraint at $-\theta_{max}$, LHC start	9	5	1
25p f_1 constraint at $-\theta_{max}$, sym. start	14	1	-

From the results in table 6.2, it can be concluded that for the 13 parameter reinforced cartwheel, the global minimum is generally found for each starting point, although sometimes 600 function evaluations were not enough. For the 25 parameter reinforced cartwheel, it is likely that local minima exist. Choosing symmetrical starting points and defining the objective function in one stroke direction while constraining the other both improve the odds of finding the global optimum. It should be noted that even though the runs of the 25 parameter reinforced cartwheel with the frequency of the full stroke as objective function resulted in finding the global optimum six times within the defined limits of 10^{-2}, the optima of the runs with the added frequency constraint at one of the maximum rotations, were orders of magnitude closer to each other. The median convergence rates of both models, using the SQP and quasi-Newton algorithm are also plotted in Figure 6.7. For this figure, the series with symmetrical starting points was used. It is clear that the SQP algorithm converges faster and closer to the optimum, especially for the 25 parameter reinforced cartwheel. At a certain point in the optimization process, the algorithm started to jump back and forth between a few points close to the optimum.



(a) 13 parameter reinforced cartwheel

(b) Asymmetrical 25 parameter reinforced cartwheel

Figure 6.7: Median convergence out of 15 runs, of the 13 and 25 parameter reinforced cartwheels. The dashed lines indicate the range excluding the two best and worst runs.

6.2.3 Analysis of local optima

For the 25 parameter reinforced cartwheel, multiple optimization runs did not result in convergence to the global optimum. Three of the local optima found were encountered with at least two different starting points. This is a strong indication that these are actual local optima. The corresponding models of the local optima found are shown in figure 6.8.

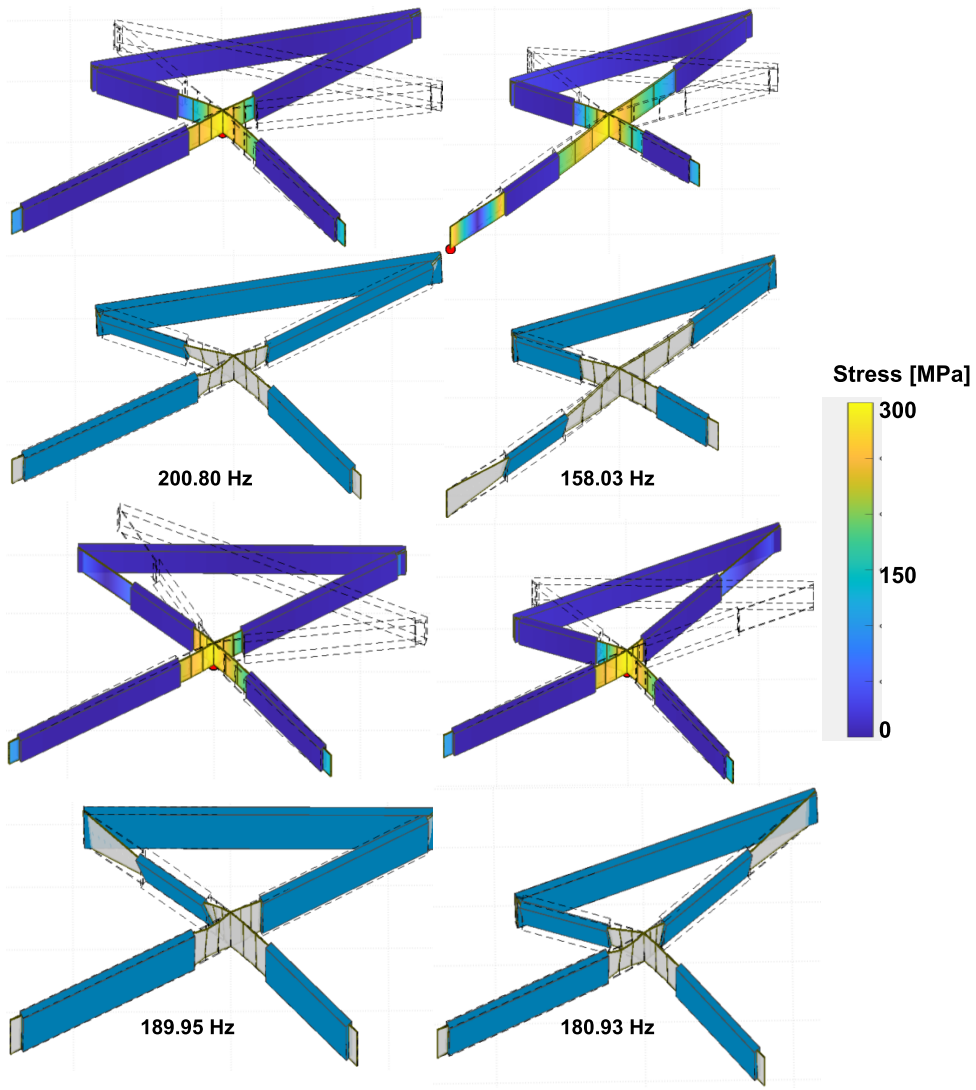
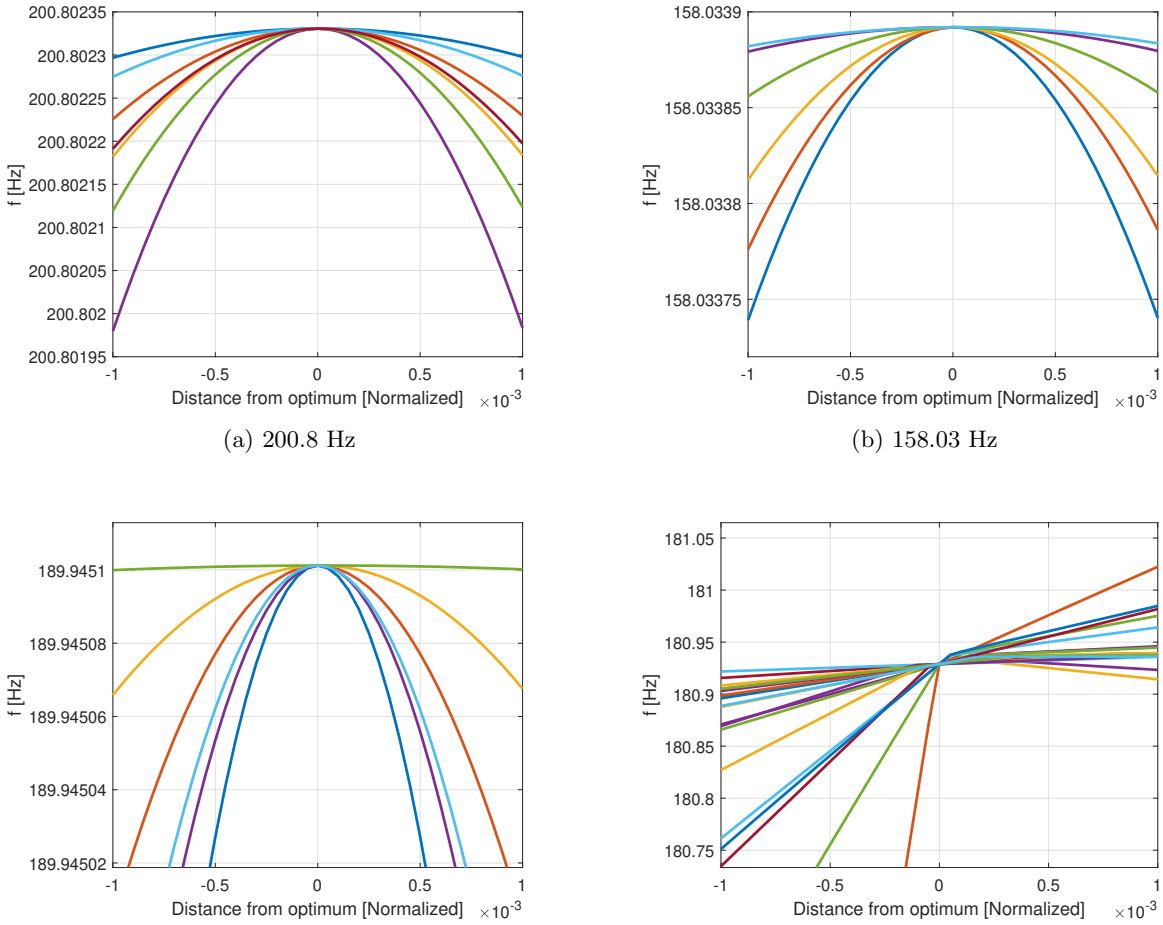


Figure 6.8: Four of the presumed global and local optima, of the asymmetrical 25 parameter reinforced cartwheel. The stress and first eigenfrequency modes at the maximum rotation are shown.

To show that these designs are indeed local optima, the optimality conditions are checked. The designs with a maximum frequency of 200.80 Hz, 189.95 Hz, and 158.03 Hz, all yield first and second order optimality, with the gradients in feasible directions close to zero and positive definite Hessian matrices in all feasible directions. The design with a frequency of 185.93 Hz, does not yield optimality. In this design, the first two eigenfrequencies are nearly identical. Convergence to this point is presumably caused by the different frequency modes taking over, causing a non-smooth objective function. The Hessian matrix calculated for checking the optimality conditions at this point is also not reliable, since the Jacobian and Hessian of the objective function are discontinuous at the location where different frequency modes take over. A sweep of the first eigenfrequency in all feasible directions is plotted in figure 6.9. This figure visually confirms that the first three designs are indeed converged to local optima, while the fourth is converged to a point where the frequency mode changes.



(a) 200.8 Hz

(b) 158.03 Hz

(c) 189.95 Hz

(d) 180.93 Hz

Figure 6.9: Parameter sweeps near converged points in all feasible directions, for the models shown in figure 6.8

Since local minima can occur, a multi-start is necessary to find the global optimum with some level of confidence. However, the odds of finding local optima can be reduced. It is wise to start the optimizing process with as few design variables as makes sense. The number of design variables can be increased gradually. With more design freedom, the optimum should always be identical or better compared to the model with fewer design variables. Stricter bounds can also reduce the chance of finding a local optimum, since it reduces design freedom. The bounds can often be stricter defined based on the results of the optimization with fewer design variables.

6.2.4 Convergence benchmark

As seen in the convergence plots, the SQP algorithm converges very well for the designs in the entire tested range of different numbers of design variables. The median number of function evaluations required to converge within 10^{-3} Hz of the best feasible solution found, is shown in figure 6.10. The data points were fitted with a second order polynomial, showing a clear trend. This graph can serve as a benchmark for the convergence rate that can be achieved for any number of design variables.

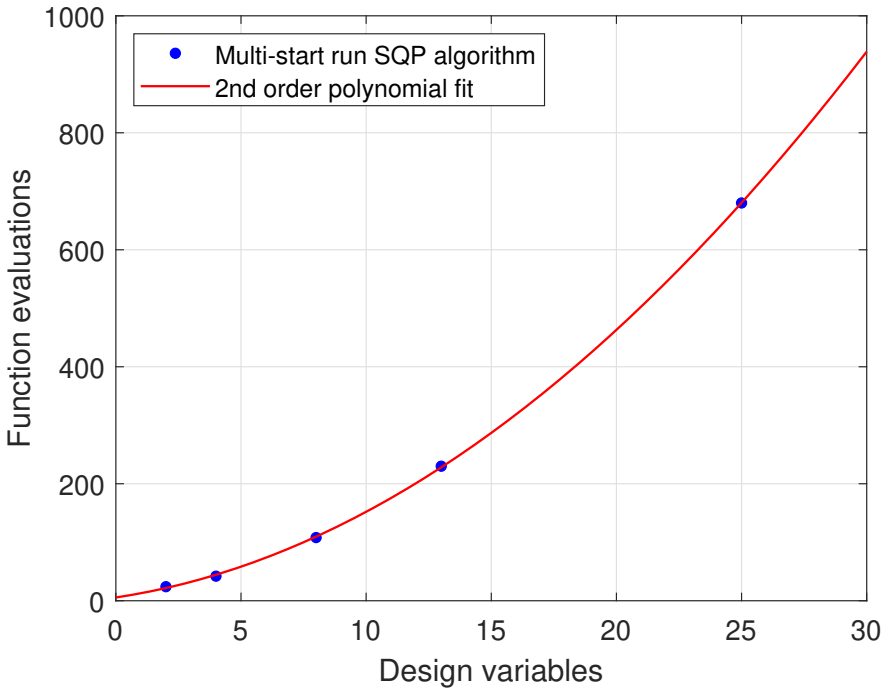


Figure 6.10: Median number of function evaluations required to converge within 10^{-3} Hz of the best feasible optimum found, using the SQP algorithm. The polynomial used is: $y = 0.8231x^2 + 6.412x + 5.574$

Chapter 7

Conclusion and recommendations

7.1 Conclusion

In chapter 4, the viability of gradient-based optimization of flexure mechanisms was demonstrated. Appropriate scaling of the objective function and normalization of the design space and constraints were shown to benefit the convergence rate. A finite difference step size of 10^{-6} in the normalized design space, was demonstrated to work well for the algorithms used in the MATLAB function fmincon, after increasing the accuracy of design parameters in the SPACAR input files generated by SPACAR Light. For the Newton's method and quasi-Newton's method a larger step size of 10^{-4} was used. These step sizes are problem and variable dependent and should be one of the first things to analyze in case of unsatisfying optimization results. The comparison of convergence rates of the different algorithms confirmed that gradient-based algorithms converge much faster than the Nelder-Mead algorithm, especially with an increasing number of design variables. Of all the tested algorithms, the sequential quadratic programming algorithm performed best.

In chapter 5, it was shown how discontinuous differentiable constraints and objective functions can occur, which can obstruct gradient-based optimization. This problem for the stress constraint was solved by using the Von Mises stress at different locations as separate constraints. A method for solving this problem for the natural frequency, caused by different modes taking over as the lowest frequency within the design space, was proposed, but not implemented yet.

In chapter 6, it was demonstrated that the developed optimization method is able to successfully optimize a flexure mechanism with up to 25 design parameters. It was also shown that increasing the design freedom of the mechanism can benefit the performance of the mechanism. The occurrence of local minima within the design space was also demonstrated for the 25 parameter reinforced cartwheel, which implies that a multi-start optimization run is required to find the global optimum with some level of confidence.

Overall it can be concluded that gradient-based optimization of flexure mechanisms modeled in the SPACAR software is viable and enables optimization with significantly more design variables compared to the Nelder-Mead algorithm. For gradient-based optimization, one of the most important considerations is to prevent discontinuous differentiable objective and constraint functions.

7.2 Recommendations

A major advantage of gradient-based algorithms over many other optimization algorithms was not discussed yet. All the function evaluations that have to be performed for determining a gradient vector or even a Hessian matrix can be done simultaneously. This is not possible for an algorithm such as Nelder-Mead, where all function evaluations depend on the outcomes of the previous function evaluations, which does not allow for parallelization. Currently, the Matlab parallel computing toolbox supports up to 512 simultaneous running processing cores. If enough processing cores are available, the run-time of an optimization with a large number of design variables can be reduced enormously. Parallel computing was not used in this research, but it is definitely recommended to implement this in future optimization projects.

It is recommended to model or keep track of the different frequency modes, to prevent the occurrence of a discontinuous differentiable objective function. This cannot be directly applied to the optimization in fmincon, since only one function can be used as objective function input. In a custom implementation of an optimization algorithm, such as the quasi-Newton method, modeling multiple frequency modes separately can be utilized.

There might be a workaround to utilize the separation of frequency modes in fmincon. The frequency modes that are the lowest somewhere within the design space could be identified with a parameter sweep. If this is a low number of modes, it could be viable to perform separate optimization runs, each time with one of the modes as the objective function, while using the other modes as constraints.

Bibliography

[1] J. B. Jonker and J. P. Meijaard, "SPACAR — Computer Program for Dynamic Analysis of Flexible Spatial Mechanisms and Manipulators," *Multibody Systems Handbook*, pp. 123–143, 1990.

[2] D. H. Wiersma, S. E. Boer, R. G. Aarts, and D. M. Brouwer, "Large stroke performance optimization of spatial flexure hinges," *Proceedings of the ASME Design Engineering Technical Conference*, vol. 6, pp. 115–124, aug 2012.

[3] D. H. Wiersma, S. E. Boer, R. G. Aarts, and D. M. Brouwer, "Design and Performance Optimization of Large Stroke Spatial Flexures," *Journal of computational and nonlinear dynamics*, vol. 9, no. 1, pp. 011016–011025, 2013.

[4] M. Naves, D. M. Brouwer, and R. G. Aarts, "Building block based spatial topology synthesis method for large stroke flexure hinges," *Journal of mechanisms and robotics*, vol. 9, p. 041006, may 2017.

[5] M. Naves, "Design and optimization of large stroke flexure mechanisms," 2021.

[6] V. J. Torczon, "Multidirectional search: a direct search algorithm for parallel machines," 1989.

[7] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, pp. 308–313, jan 1965.

[8] K. I. McKinnon, "Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point," *SIAM J. Optim.*, vol. 9, no. 1, pp. 148–158, 1998.

[9] P. Kuresangsai and M. O. T. Cole, "Kinematic modeling and design optimization of flexure-jointed planar mechanisms using polynomial bases for flexure curvature," *Mechanism and Machine Theory*, vol. 132, pp. 80–97, 2019.

[10] N. Lobontiu and E. Garcia, "Analytical model of displacement amplification and stiffness optimization for a class of flexure-based compliant mechanisms," *Computers Structures*, vol. 81, pp. 2797–2810, dec 2003.

[11] J. R. R. A. Martins and A. Ning, "Engineering Design Optimization," 2021.

[12] "Constrained Nonlinear Optimization Algorithms - MATLAB Simulink - MathWorks Benelux."

[13] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[14] "Find minimum of constrained nonlinear multivariable function - MATLAB fmincon - MathWorks Benelux."

[15] J. A. Haringx, "The cross-spring pivot as a constructional element," *Applied Scientific Research*, vol. 1, pp. 313–332, dec 1949.

[16] D. M. Brouwer, J. P. Meijaard, and J. B. Jonker, "Stiffness analysis of parallel leaf-spring flexures,"

Appendix A

Scaling of a compliance objective function

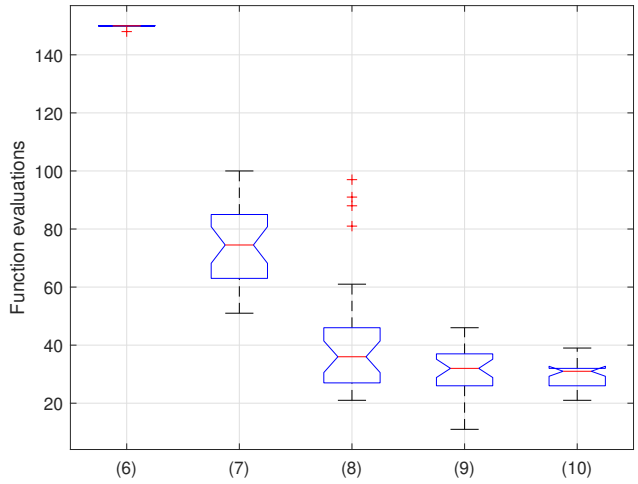


Figure A.1: Function evaluations required to converge within 10^{-8}, of the best solution. Note that the maximum number of function evaluations was set at 150. The compliance in node 3, see Figure 4.1a, in y-direction C_{yy} was minimized.

Table A.1: Objective function corresponding to box plots in figure A.1

Box nr.	Description
(6)	$C_{yy}(x)$
(7)	$\sqrt[3]{C_{yy}(x)}$
(8)	$C_{yy}(x)/C_{yy}(x0)$
(9)	$\sqrt[3]{C_{yy}(x)/C_{yy}(x0)}$
(10)	$log(C_{yy}(x))$

Appendix B

Forward differences of the constraint functions

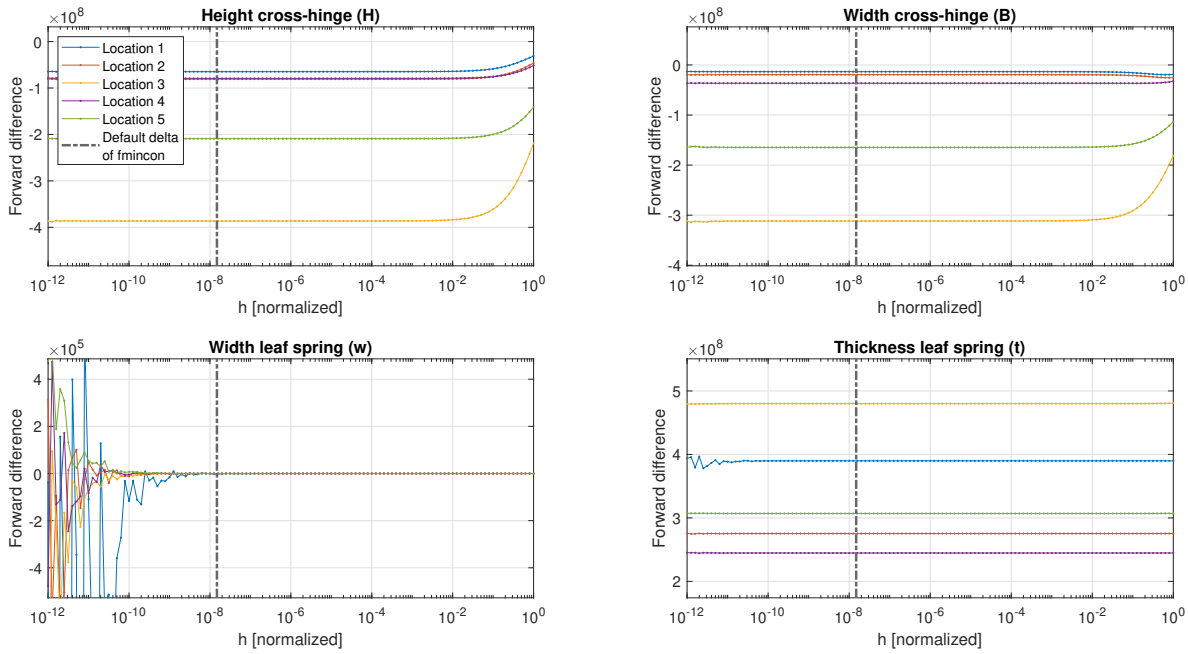


Figure B.1: Delta sweep of forward differences of the maximum Von Mises stress to the four normalized design variables of the four parameter cross-hinge

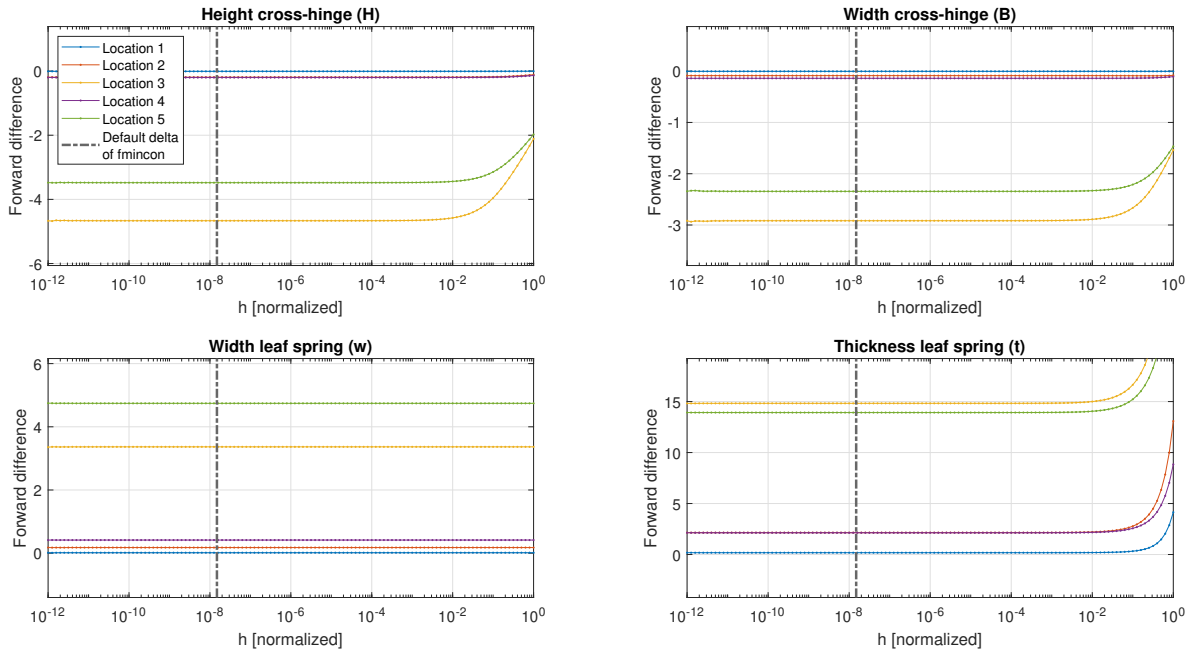


Figure B.2: Delta sweep of forward differences of the actuation moment to the four normalized design variables of the four parameter cross-hinge

Appendix C

Optima

2p	4p	8p	6p	13p	25p
0.0730013	0.0730013	61.0702	0.30000	0.30000	0.30000
0.0004417	0.0500000	52.7010	0.44772	0.30163	0.30000
	0.0750000	50.0000	0.25617	0.30000	0.30187
	0.0004417	50.0000	0.95000	0.51846	0.30000
		75.0000	50.00000	0.24128	0.36420
		75.0000	29.92528	0.24307	0.30000
		0.4348		0.95000	0.71884
		0.4348		0.95000	0.42674
				50.00000	0.17177
				50.00000	0.28201
				34.88615	0.15675
				20.00000	0.34408
				75.00000	0.95000
					0.93702
					0.95000
					0.95000
					70.00000
					40.00000
					70.00000
					40.00000
					32.97340
					38.07524
					20.00000
					22.23413
					75.00000

Figure C.1: The optimal design variable vectors of the two, four, and eight parameter cross-hinge and the 6 and 13 reinforced cartwheel and finally the 25 parameter asymmetrical reinforced cartwheel with $\pm\theta$. For the corresponding optimization problems, see Chapters 4, 5, and 6.