Challenges and Solutions to Arc-Length Controlled Structural Shape Design Problems

Johann M. Bouwer[a*], Schalk Kok[a], Daniel N. Wilke[a]

[a] Department of Mechanical Engineering, University of Pretoria, Pretoria, South Africa

Johann.M.Bouwer@gmail.com[*]

Challenges and Solutions to Arc-Length Controlled Structural Shape Design Problems

This paper presents a general and reliable optimization scheme that performs shape optimization of snap-through structures in order to match a target load-deflection curve. The developed optimization scheme is insensitive to the complexity of the target or simulated load-deflection paths. The objective function for the optimization problem is the mismatch between the simulated load-deflection curve and the target load-deflection curve. The simulation of the snap-through structures typically requires the implementation of the Arc Length Control (ALC) method. This path-following algorithm allows the solution of problems that typical non-linear finite element simulations with load or displacement control are unable to simulate fully. Our implementation of this algorithm automatically adjusts the solution step size during the simulation, to enhance the algorithm's ability to solve highly non-linear problems. This adaptive stepping results in the presence of discontinuities in the objective function. The discontinuities are a direct result of the changing number of solution steps in the simulation, and the location of these solutions along the load-deflection curve. Conventional wisdom dictates that these discontinuities must be eliminated by using a small constant step size, or the discontinuous objective functions must be solved by zero-order methods. However, we opt to solve the optimization problems using gradient-only optimization. Gradient-only optimization defines non-negative gradient projection points as the solution to the optimization problem. We therefore require reliable sensitivity information to isolate these points. Due to the presence of discontinuities in the objective function, the implemented sensitivity procedure is analytical as finite differences can return incorrect results, unless complex-step finite-differences are employed. Numerical examples demonstrate that the discontinuities in the objective function render conventional gradient based approaches ineffective. This is due to the use of function value information to determine if a local minimizer was found. Function value based methods can misinterpret discontinuities as minima and hence terminate at poor solutions. In contrast, gradient-only algorithms are demonstrated to be insensitive to numerical discontinuities, and they locate the correct solutions reliably.

Keywords: Shape Sensitivity, Arc Length Control, Optimization, Discontinuities, Gradient-Only.

1 Introduction

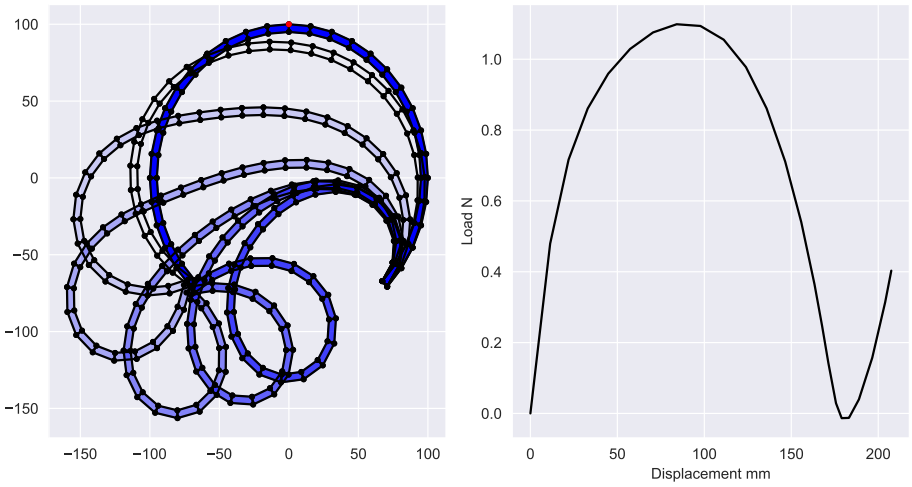
The goal of this paper is to develop a general and consistent method to complete shape optimization of highly non-linear snap-through or snap-back structures. The developed optimization scheme must be insensitive to the complexity or non-linearity present in the problem. This task involves optimizing the values of various shape parameters that describe a known snap-through structure such that the structure exhibits a desired load-deflection path.

Previous work in the task of optimizing these structures [1–5], are almost exclusively concerned with topology optimization of these structures. The majority of previous work [1–3] limit the complexity of the simulated load-deflection paths such that displacement control is adequate to fully simulate the structure's behavior. This paper on the other hand places no limit on the complexity of the simulated behaviour and implements a far more reliable simulation and optimization strategy.

The main contribution of the research completed in this paper over previously established work is the handling of complications that arise due to the complex solution strategy required to always be able to solve these highly non-linear load-deflection paths [6]. Although discussed in more technical detail later in the paper, the solution strategy implemented to solve these load-deflection paths requires the designer to surrender control of where the simulation returns known solution locations. To overcome this lack of control, researchers needed to implement complex interpolation strategies to find the solutions at standard locations [4, 5]. The results in this paper show that these complex interpolation strategies are a redundant ingredient in an already complex design problem.

Figure 1 shows an example snap-through structure, commonly known as the Deep Semi-Circular Arch [7], as well as its associated load-deflection curve. Although this paper exclusively makes use of the Deep Semi-Circular Arch for demonstration purposes there are many other known snap-through structures. Leahu-Aluas and F. Abed-Meraim [8] offer many examples of snap-thorough structures as benchmark buckling problems. In addition, although the solution strategy proposed in this paper is applied to a shape optimization problem, it can also be applied to solve topology optimization problems where the structural analysis is performed by the arc length control algorithm.

The simulation of these structures solve for a discrete number of positions along the load-deflection path, typically using some variant of the Arc Length Control (ALC) algorithm [9]. But the real challenge is not analyzing a given structure to obtain the load-deflection curve, it is rather that the desired load-deflection curve is known and the structure that exhibits this behaviour is unknown. Therefore an inverse problem needs to be solved that minimizes the discrepancy between the desired



(a) (b)

Figure 1: (a) The Deep Semi-Circular Arch as an example snap-through structure with the loaded node indicated in red on the initial undeformed state, and (b) the associated load-deflection curve of the loaded node.

load-deflection curve and a simulated curve. Since subtle changes in the structure's shape parameters could drastically change the resultant load-deflection curve, automatic arc length adjustment is used to ensure that the analyses can be performed for any design.

However, an unintended consequence of automatic arc length adjustment is that it creates discontinuities in the objective function that quantifies the discrepancy between the target load-deflection curve and the simulation counterpart. Note however that these discontinuities are purely a numerical artefact: their presence should be ignored by any algorithm that seeks to find the optimal solution. Figure 2 shows such a discontinuous objective function for a simple two variable load path optimization problem. It is evident that classical gradient based optimization algorithms will struggle to navigate this objective function landscape, often getting stuck at the discontinuities. A lesser known strategy is to locate non-negative gradient projection points (NN-GPPs) as defined in the gradient-only optimization problem [10]. This strategy was developed specifically to optimize discontinuous objective functions, where the discontinuities are numerically induced [10–16]. Gradient-only optimizers developed to locate NN-GPPs [10–13] have been demonstrated to solve problems of this type reliably and efficiently.

The structure of this paper is as follows. Firstly, following Snyman and Wilke [10], Section 2 presents three formulations to define the solution to an optimization problem. These formulations are the function minimizer, optimality criteria (necessary and sufficiency conditions) and NN-GPPs. The

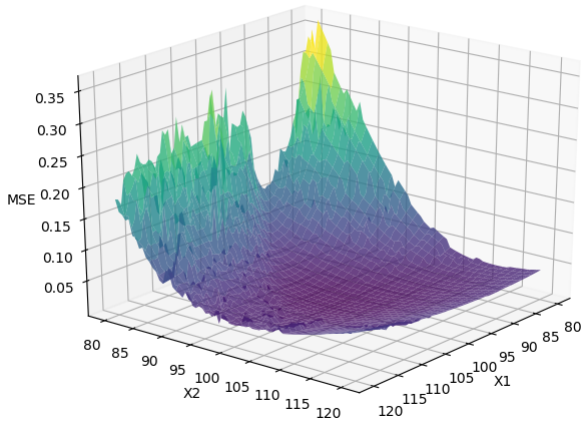


Figure 2: Example of a discontinuous objective function for highly non-linear load path optimization

importance of NN-GPPs as a reliable optimizer formulation for piece-wise discontinuous functions is highlighted. A brief explanation of the Arc Length Control method is completed in Section 3. From the ALC description the source of the discontinuities depicted in Figure 2 are detailed in Section 4. Section 5 verifies the analytical sensitivities required for the gradient-only optimizers. The paper concludes with a systematic numerical investigation in Section 6, which highlights the inverse shape optimization problem to recover a given load-displacement path for a selected snap-through structure [8]. The number of shape parameters are increased from two to eight, allowing the behaviour and performance of the optimization algorithms to be investigated. The selected algorithms are the function minimization algorithm, Sequential Least Squares Programming (SLSQP) [17], the gradient-only sequential spherical approximation method (GOSSA) [10], and a Modified Subgradient method [10, 11].

2 Three Formulations for Optimizers

The problem considered in this paper, which is to reduce the discrepancy between two curves, is an unconstrained optimization problem. Three formulations exist to define solutions to this problem, namely function minimization (Section 2.1), optimality criteria (first and second-order conditions) (Section 2.2) and NN-GPPs (Section 2.3).

2.1 Formulation 1 - Function minimizer

In general the unconstrained optimization problem is expressed as

$$\underset{w.r.t\ \mathbf{x}}{\text{Minimize}}\ F(\mathbf{x}),\ \mathbf{x} = [x_1, x_2, ..., x_n]^{\mathrm{T}} \epsilon \mathcal{R}^n, \tag{1}$$

where $F(\mathbf{x})$ is a scalar objective function to be minimized with respect to the column vector $\mathbf{x}$, which consists of real and continuous values. These values are referred to as design variables or, in the case of shape optimization, shape parameters such as lengths, thicknesses, or radii. In this paper the shape parameters consist of radii at specified angles.

The first option to define the optimal solution, $\mathbf{x}^*$, uses objective function values. The definition is

$$F(\mathbf{x}^*) \leq F(\mathbf{x}) \text{ for all } \mathbf{x}. \tag{2}$$

As it is typically impossible to evaluate the function $F(\mathbf{x})$ for all values of $\mathbf{x}$, the solution $\mathbf{x}^*$ is defined as a strong local minima if the above condition is met for some local region in multi-dimensional space that defines $\mathbf{x}$. The optimization problem at this point has been defined as a minimization problem. Gradient based algorithms are typically considered as the most efficient to solve this optimization problem. A conventional gradient based optimization algorithm will use the gradient information to determine a direction to update the proposed optimum solution until the minimization criterion, $F(\mathbf{x}^*) \leq F(\mathbf{x})$, is met. This implies that *only* the value of the function $F(\mathbf{x})$ is considered when defining an optimal solution. Considering the objective function landscape in Figure 2, the presence of discontinuities will result in premature convergence of such an algorithm. This necessitates an alternative definition of the optimal solution $\mathbf{x}^*$.

2.2 Formulation 2 - Optimality criteria

The optimal solution to an unconstrained minimization problem can also be defined by the optimality criteria. The 1^{st} and 2^{nd} order criteria, also defined as the necessary and sufficiency condition respectively, are given by

$$\frac{dF(\mathbf{x})}{d\mathbf{x}} = \mathbf{0}, \tag{3}$$

and

$$\mathbf{x}^{\mathrm{T}} \frac{d^2F(\mathbf{x})}{d\mathbf{x}^2}\mathbf{x} > 0\ \ \forall\ \ \mathbf{x} \neq \mathbf{0}. \tag{4}$$

These criteria require that at the proposed solution the norm of the objective function gradient is zero, and the Hessian of the objective function is positive definite (the curvature in any direction is non-negative). These two conditions are sufficient to define a candidate local minimum. One can therefore define a candidate optimum position, $\mathbf{x}^*_{\mathbf{NS}}$, as any vector that results in the gradient vector to be zero. Of course there are potentially multiple solutions for which a scalar function can exhibit a zero gradient vector, such as a local maximum or a saddle point. Therefore, to define a local minimum, the sufficiency condition is required to define an optimum solution vector $\mathbf{x}^*_{\mathbf{N}}$. Algorithms that solve the optimality criteria typically require the Hessian of the objective function. Therefore these algorithms are not as popular as gradient based algorithms that only require the gradient of the objective function.

2.3 Formulation 3 - NN-GPP

The last alternative to define the optimal solution to the unconstrained minimization problem is to locate non-negative gradient projection points $\mathbf{x}^*_{\mathbf{nngpp}}$ [13]. A non-negative gradient projection point is defined as a position in the design space for which any update to the solution vector, projected onto the objective function gradient vector, is positive (or zero). These points are defined formally as a point that for every $\mathbf{u}\{\mathbf{y} \in \mathcal{R}^n, ||\mathbf{y}|| = 1\}$ there exists a real number that satisfies the condition

$$\nabla_A F(\mathbf{x}^*_{\mathbf{nngpp}} + \lambda \mathbf{u})^{\mathrm{T}} \mathbf{u} \geq 0 \ \forall \ \lambda \ \in (0, r_u], \tag{5}$$

where λ is the magnitude of the vector in the search direction.

This implies that as one approaches and passes such a point, the projected gradient changes sign from negative to positive at $\mathbf{x}^*_{\mathbf{nngpp}}$. Note that this definition of the optimal solution requires *only* the gradient information of the objective function, since the objective function values themselves are never used. This criterion is sufficient to define a local optimum. Such a definition of optimality will be able to ignore the numerically induced discontinuities present in the problem of interest, as long as the gradient information remains consistent across such a discontinuity. A visual illustration of these non-negative gradient projection points, and how they differ from the other formulations, are presented in Section 2.4.

2.4 Optimizer Characteristics

To highlight the importance to differentiate between the four optimizers, $\mathbf{x}^*, \mathbf{x}^*_{\mathbf{NS}}, \mathbf{x}^*_{\mathbf{N}}, \mathbf{x}^*_{\mathbf{nngpp}}$, we consider the four uni-variate objective functions shown in Figure 3:

- Two infinitely differentiable functions (C^∞), one being convex (Subproblem A) and the other non-convex (Subproblem C),
- a C^0 continuous function (Subproblem B), and
- a piece-wise discontinuous function (Subproblem D).

The associated derivatives are depicted in Figure 4.

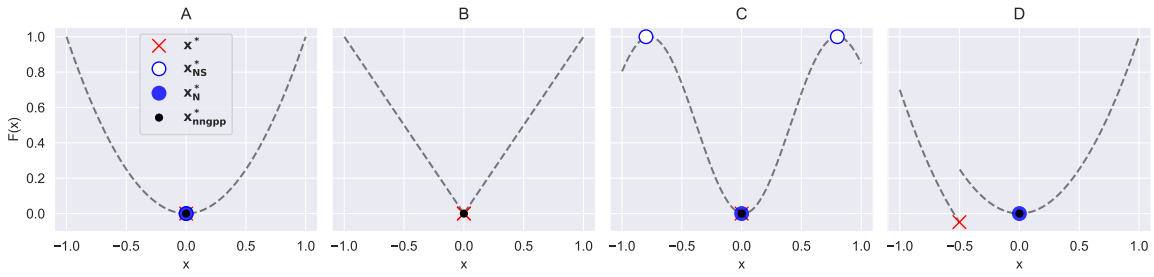


Figure 3: The Four Optimizers for Four One-Dimensional Problems

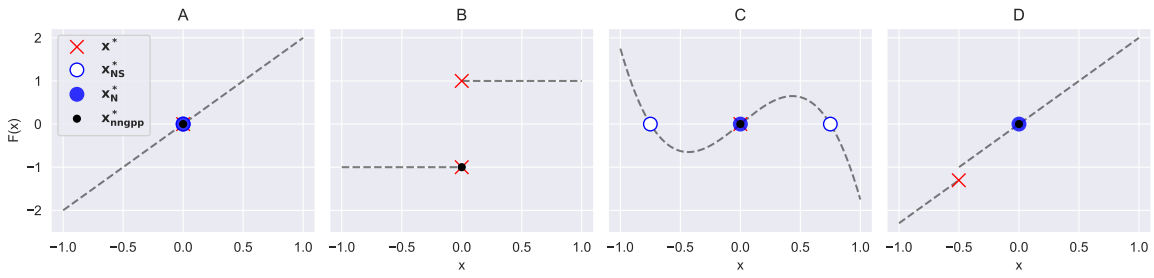


Figure 4: The Four Optimizers for Four One-Dimensional Problems in the Gradient Space

The four optimizers are equivalent for sub-problem A, the infinitely differentiable convex function shown in Figure 3. The infinitely differentiable non-convex function in sub-problem C distinguishes between $\mathbf{x}^*_{\mathbf{N}}$ and the other three optimizers, $\mathbf{x}^*$, $\mathbf{x}^*_{\mathbf{NS}}$ and $\mathbf{x}^*_{\mathbf{nngpp}}$, which are again equivalent for this problem.

The non-smooth function in sub-problem B demonstrates the differences between the two optimums $\mathbf{x}^*_{\mathbf{N}}$ and $\mathbf{x}^*_{\mathbf{NS}}$ and the remaining two optimums $\mathbf{x}^*$ and $\mathbf{x}^*_{\mathbf{nngpp}}$. The gradient plot shows that no x-position has a gradient of zero. Therefore, the two optimal positions $\mathbf{x}^*_{\mathbf{N}}$ and $\mathbf{x}^*_{\mathbf{NS}}$ do not exist.

Lastly, sub-problem D is the same as sub-problem A, but it contains a discontinuity at $x = -0.5$. This discontinuity creates a difference in the $\mathbf{x}^*$ point when compared to the other optimum points. Optimizers $\mathbf{x}^*_{\mathbf{NS}}, \mathbf{x}^*_{\mathbf{N}}$ and $\mathbf{x}^*_{\mathbf{nngpp}}$ agree exactly even for non-continuous problems.

Therefore, these simple problems demonstrate the reliable nature of the positive gradient projection points $\mathbf{x}^*_{\mathbf{nngpp}}$. These points are unaffected by local maxima, which can impact $\mathbf{x}^*_{\mathbf{N}}$, or discontinuities in either the function or gradient space which impact $\mathbf{x}^*$.

3 The ALC Algorithm

Solving the inverse shape optimization problem efficiently, in order to recover a given load-displacement curve, requires an analysis framework that makes analytical sensitivities available over the entire load-displacement path.

In this work the ALC algorithm is selected to provide the analysis. To justify this selection three possible load-deflection paths are presented in Figure 5 that may be encountered during the optimization process. Figure 5(A) shows a curve that can be solved with the ALC method, load control, and displacement control. Figure 5(B) can only be solved with displacement control or the ALC method. Finally, Figure 5(C) can only be solved with the ALC method.

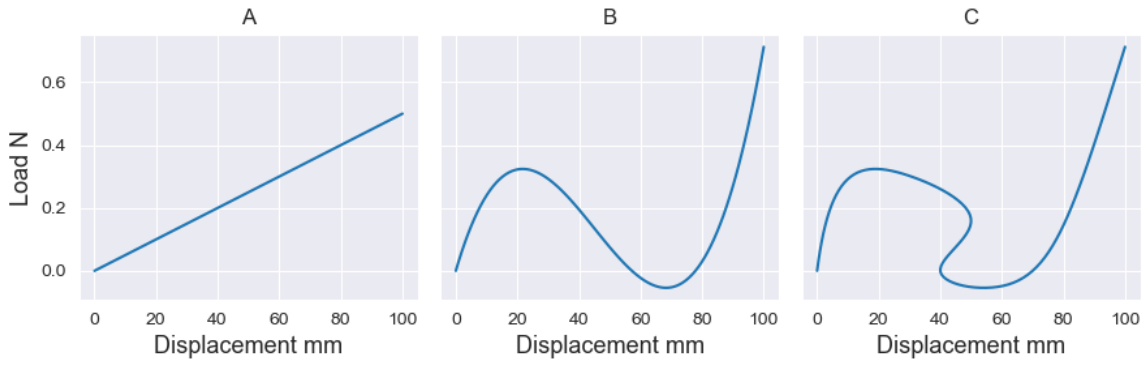


Figure 5: Three Possible Load-Deflection Curves that can be Exhibited by Structures during the Optimization Process.

Therefore, the ALC method allows for more complex behaviour to be simulated and, because of

this, will also allow for more complex target curves to be provided to the optimization problem. The implementation of the ALC method over displacement control means that the restrictions placed in previous work on this topic [1–3] are no longer needed, and a far more general load-deflection path optimization can be completed.

The ALC algorithm is a path following method that was first proposed by Riks [9]. It is often implemented when load or displacement control algorithms fail to trace the entire load path due to the highly non-linear behaviour of the simulated structure. The algorithm is implemented in a non-linear Finite Element (FE) research code in the Octave environment.

In typical non-linear FE analysis, the governing residual equation is expressed as

$$\mathbf{R} = \mathbf{K}(\mathbf{u})\mathbf{u} - \lambda\mathbf{F}, \tag{6}$$

where $\mathbf{K}$, $\mathbf{u}$, λ, and $\mathbf{F}$ denote the stiffness matrix, the nodal displacement vector, load parameter, and the load vector respectively. In the load control scheme, the goal of some iterative non-linear solver is to find the nodal displacement vector at some load parameter that reduces the governing residual, Equation (6), to $\mathbf{0}$ within some desired tolerance.

This iterative solver typically takes the form of Newton's method, where the gradient of Equation (6) with respect to the nodal displacement vector is computed. A problem with this solution strategy arises when the load path has a limit point, as Newton's method cannot fully trace the curve past this point.

The ALC algorithm proposes a solution to this problem by adding an additional constraint equation,

$$L^2 = \blacktriangle\mathbf{u}^{\mathrm{T}}\blacktriangle\mathbf{u} + \psi^2\blacktriangle\lambda^2\mathbf{F}^{\mathrm{T}}\mathbf{F}. \tag{7}$$

Here, L denotes the prescribed arc length, $\blacktriangle\mathbf{u}$ is the total displacement vector update for the current load step, $\blacktriangle\lambda$ is the total load parameter increment for the current load step, and ψ is some non-dimensional scale factor. The scale factor scales the load increment to a more appropriate value such that the load increment and displacement increment contribute similarly to the constraint equation. This ψ variable is often set to zero, typically referred to as spherical constraint, but if selected appropriately can be beneficial to solution times. For the remainder of this paper, it is assumed the load

vector $\mathbf{F}$ is a single point load with unit magnitude. Therefore Equation (7) can be simplified to

$$L^2 = \blacktriangle \mathbf{u}^{\mathrm{T}} \blacktriangle \mathbf{u} + \psi^2 \blacktriangle \lambda^2. \tag{8}$$

This constraint equation implies that both the displacement vector update and load parameter increment are now unknowns. The prescribed arc length, L, and ψ variables are problem dependant hyper-parameters. Typically, the ALC algorithm requires the prescribed arc length, L, and then an accumulated arc length that terminates the simulation once reached. The selection of an appropriate prescribed arc length and accumulated arc length is problem specific, and often requires some iteration and experimentation to find a combination that yields satisfactory results.

In order to ensure that the ALC algorithm can compute the load path for all designs, the prescribed arc length can be automatically adjusted based on one of two scenarios. Firstly, if there is no intersection between the prescribed arc length and the equilibrium path, the prescribed arc length is decreased by some factor until an intersection is found. Secondly, if the intersection is not found within some set number of iterations, the prescribed arc length is also decreased by some factor and the equilibrium iterations begin again. This automatic prescribed arc length adjustment increases computational efficiency as the algorithm can now take large solution steps when the load path behaves close to a linear fashion, and then adjust to take smaller solution steps when the load path behaviour is more complex. The automatic prescribed arc length adjustment heuristic is central to the solution process followed in this paper.

The inclusion of the arc length constraint creates a new system of equations that needs to be solved with some iterative solver. The reader is referred to literature for solution strategies for this new system of equations [18–20].

Structures that exhibit highly non-linear behaviour, such as the snap-through behaviour presented in this research, typically exhibit this behaviour when loaded in bending. Therefore the 4-noded assumed stress element [21] is used for the simulations in this research. This element type is exact in bending for linear elasticity, and remains highly accurate in bending for non-linear elasticity. For a more detailed description of these elements refer to Appendix A.2.

4 Discontinuities in the Design Problem

The automatic prescribed arc length adjustment creates discontinuities in any calculation that compares one load-deflection curve to another. This is because the number and locations of the discrete solution positions along the load path can differ suddenly due to an infinitesimal change in the structure shape. This can be shown by considering a simple design problem. This problem is taken from a proposed set of benchmark buckling problems [8]. The Deep Semi-Circular Arch benchmark problem is converted into a design problem where the radius of the arch is the shape parameter to be optimized. Figure 6 shows an example mesh for an arch with constant radius. The arch is clamped at the right edge and pinned on the left. Figure 7 shows the resultant displacement overlay and load displacement curve.

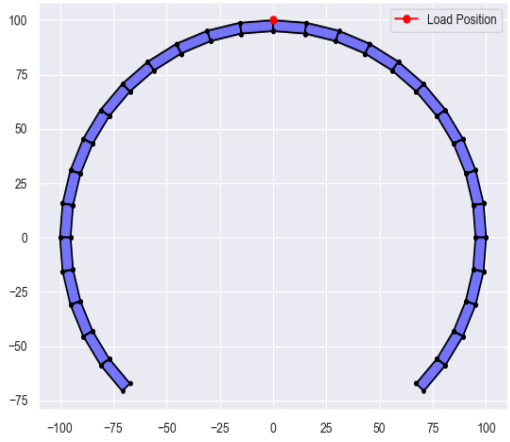


Figure 6: Symmetrical Deep Semi-Circular Arch Mesh

As the goal is to find a structure that exhibits the same load path as some specified load path, an objective function that quantifies the difference between two curves needs to be established. In general these load-deflection curves need not be functions, such as the curves presented in Figures 5(A) and (B), which can be solved with displacement control as the load can be expressed as a function of the displacement. Instead the load-deflection curves are represented as parametric curves, such as Figure 5(C), where the accumulated arc length is chosen as the independent variable and therefore require the implementation of the ALC algorithm. However, one complication is that both curves are represented by a finite number of points on the curve. Also, the solution points are unlikely to be

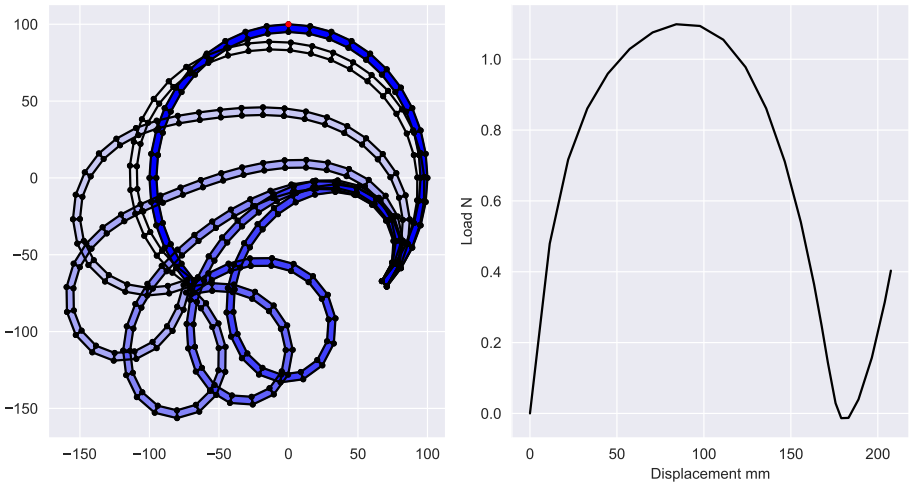


Figure 7: Displacement Overlay and Load Displacement Curve for a Symmetrical Deep Semi-Circular Arch.

available at the identical accumulated arc lengths used to represent the target curve.

Previously, this complication is either avoided or overcome with two broad categories of methods. The first category involves limiting the complexity of the prescribed load-deflection path. Previous research [1–3] require the desired load-deflection behaviour to be non-parametric. This means that the problems must be such that the load can be expressed as a function of the displacement, such as in Sub-Figure 5B. The desired path is then described with a set number of load values at predetermined displacement values. This method eliminates the aforementioned arc length adapting complication, as a simple displacement control solution scheme can be implemented and the ALC method is then no longer needed. However, this method cannot be used in general to solve more complex load-deflection path optimization problems where the load cannot be expressed as a function of the displacement. Therefore, this method requires a restriction of the design space in order to avoid more complex load-deflection paths.

The second more general method implemented by Bruns et al. [4, 5] handles this problem by constructing a linear interpolation function that allows the designer to find the deformed state of the structure at accumulated arc length locations other than the returned simulation locations. This method requires the previously deformed state $\mathbf{q}$ at some accumulated arc length $\mathbf{a}$ and the current deformed state q^* at some greater accumulated arc length value a^*. Then all the degrees of freedom in the mesh are interpolated linearly to some deformed state $\overline{q}$ at some intermediate accumulated arc length value $\overline{a}$ while ensuring that the ALC constraint condition is still met. This method can be

computationally expensive for meshes with a large number of degrees of freedom. Therefore, this paper proposes a far simpler interpolation strategy whereby only interpolation functions for the parametric load and displacement curves are needed instead of interpolation functions for each degree of freedom in the mesh.

The analyst must still choose between two interpolation options, namely for each iteration to:

1. interpolate the target points to the simulated points at the same accumulated arc lengths, or
2. interpolate the simulated points to target points at the same accumulated arc lengths.

For the rest of the paper interpolating the target curve is referred to as objective function **A** and interpolating the simulated curve as objective function **B**.

Interpolation between target points or simulated points is required each time the objective function is called, to ensure that the error terms are calculated at the same accumulated arc length. For the sake of simplicity, only linear interpolation is considered in this paper. Figure 8 depicts example load curves that illustrate both interpolation options graphically.

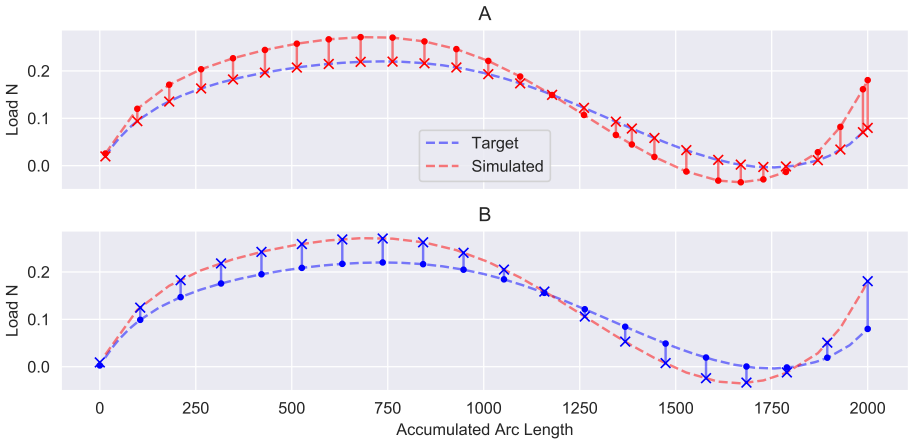


Figure 8: Comparison of two linear interpolation schemes with points denoting known locations and crosses interpolated locations. A - Interpolating the target curve, B - Interpolating the simulated curve.

Interpolating to the same accumulated arc lengths then allows for the computation of the mean square error,

$$MSE = \frac{1}{N}\sum_{i}^{N}(\lambda_i^{\mathrm{T}} - \lambda_i^d)^2 + \frac{1}{N}\sum_{i}^{N}(u_i^{\mathrm{T}} + u_i^d)^2, \tag{9}$$

between points on the target curve, λ_i^{T} and u_i^{T}, and points on the simulated curve of the current design, λ_i^d and u_i^d, at identical accumulated arc lengths. Here N is the number of solution steps taken

to solve the curve (objective function **A**) or the number of points used to describe the target curve (objective function **B**).

Sampling the proposed objective functions for the Deep Semi-Circular Arch 200 times between arc radii from 80mm to 120mm, where the load path at 100mm is the target curve, computes Figure 9. All the simulations were completed with the same prescribed arc length as well as a fixed accumulated arc length that terminates the solution once reached. The prescribed arc length is decreased by a factor of $\sqrt{2}$ when more than nine iterations are required to find a solution.

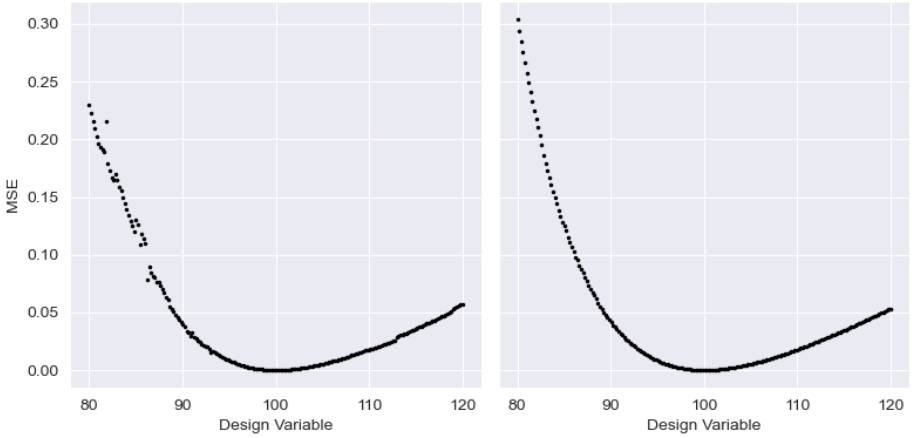
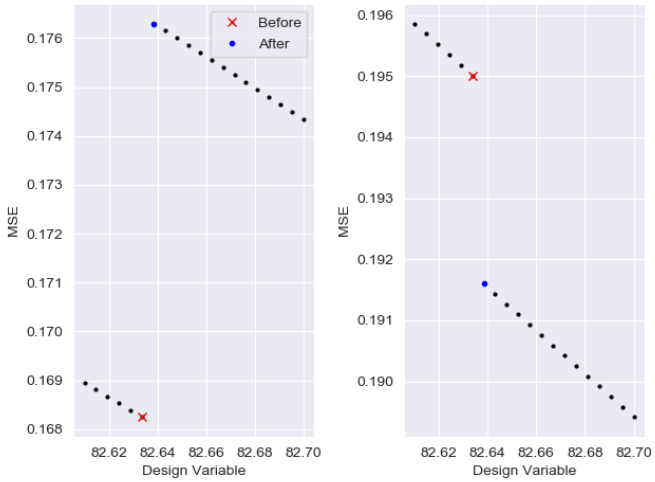


Figure 9: One-dimensional objective functions sampled 200 times between arc radii of 80 and 120 mm. Sub-Figure A shows objective function **A**, while Sub-Figure B shows objective function **B**.

Objective function **A** contains numerous large discontinuities even for this simple one-dimensional problem. Objective function **B** lessens the severity of the discontinuities, but, there still seems to be a few small discontinuities present in the problem. To gain a better understanding of the discontinuities present, in either objective function, Figure 10 presents an explicit line search for a small variation in the design variable at the same shape.

Therefore, both interpolation schemes create a discontinuity at the same location in the design space. The magnitude and direction of this discontinuity may differ, but neither interpolation scheme can avoid them from manifesting in the objective function. The underlying source of these discontinuities is the automatic arc adjustment that creates a change in the number of discrete points along the curve as well as the coordinates of these points. This can be shown by plotting the solved load paths before and after the discontinuity in Figure 11.

These two load paths in Figure 11 are nearly identical but the ALC method returns significantly different discrete locations along these curves. Both the arches require arc length adjustments dur-



(a) (b)

Figure 10: Dense line search at the discontinuity around 82.6mm for the same structure, for (a) objective function **A** and (b) objective function **B**. Note the reduction in the magnitude of the discontinuity in objective function **B** compared to objective function **A**.

ing the simulation but the size and number of these adjustments differ. This creates sudden and discontinuous changes in the objective function for either interpolation scheme.

For objective function **A** the source of these discontinuities is shown in Figure 12, where the load and displacement curves before and after the discontinuity are shown in the arc length space. The change in the accumulated arc positions means the error calculation takes place at locations that could be further away from the target curve, creating an increase in the error calculation, even though these are still discrete locations on the same curve.

In the case of objective function **B** it is not as obvious what the source of these discontinuities are. The error calculations consistently contain the same number of points and these points are consistently at the same accumulated arc length coordinates. Therefore, an analyst can easily assume that the objective function will contain no discontinuities, but, as can be seen in Figures 9 and 10, this is not the case.

Figure 13 explains the source of these discontinuities, where an example curve in the accumulated arc length domain is shown. When the locations of the solved positions along the curve change, the interpolation can suddenly return significantly different results. Assume that a structure is analyzed, resulting in 5 solutions along the load path (the blue plus signs 1 to 5). An infinitesimal adjustment in the structure could trigger the prescribed arc length adjustment, resulting in 4 solutions along the load path (the green plus signs 1 to 4). The objective function is computed at different locations

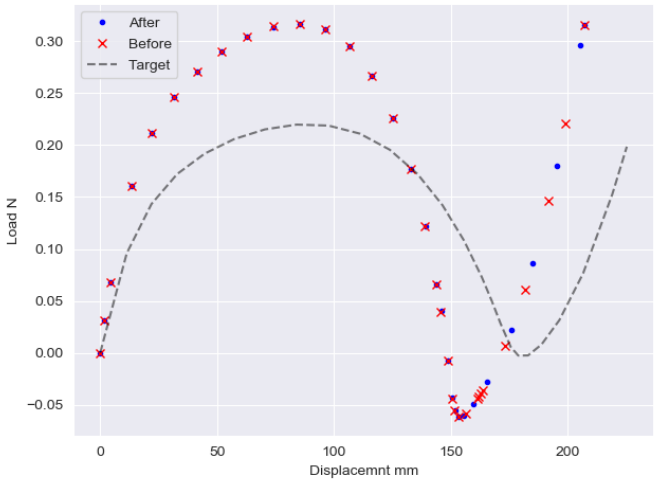


Figure 11: Discrete points on the solved load paths before and after the discontinuity.

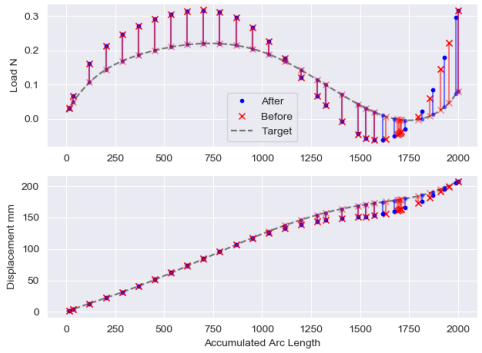


Figure 12: Discrete solved points before and after the discontinuity in the accumulated arc length space for the load and displacement curves.

altogether, indicated with the solid blue dots. Since none of the solutions are available at the same locations, linear interpolation (dashed lines) is used to interpolate the solution coordinates to the required coordinates. Notice that the error contributions at locations i and $i+1$ undergo a step change due to different points along the same curve being used to compute the error contributions.

Although Bruns et al. [4] make no mention of discontinuities due to their interpolation scheme in the solved optimization problem, it is the authors' opinion that there is anecdotal evidence of the presence of these discontinuities. Bruns et al. [4] report an improvement in the optimum result after decreasing the step size of the optimization process and attribute this to a local minimum in the objective function. From Figure 13 it can be seen that a smaller step size will reduce the magnitude of the discontinuities in the objective function, therefore, the authors believe that what is attributed to the

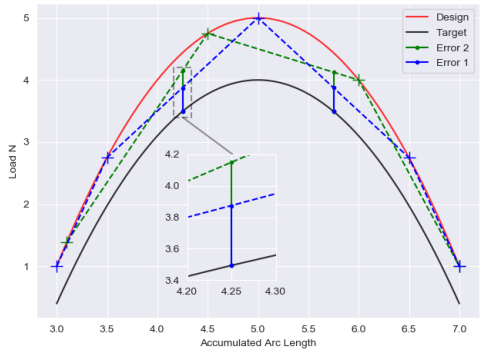


Figure 13: Illustration of how objective function **B** is computed, and how it can result in discontinuities.

optimiser bypassing a local minimum is in fact the optimisers initially misinterpreting a discontinuity as a minimum. When the step size was reduced the severity of the discontinuity was likely reduced such that the optimizer was then able to bypass it. Therefore, even the complex interpolation strategy implemented by Bruns et al. [4, 5] may still result in discontinuities in the objective function. This is consistent with Bruns et al. [5] reporting large oscillations in the objective function, referred to as "chattering", and stating that efforts to avoid these oscillations were ineffective.

To summarize, the differences in the number of solution points and their respective locations, combined with subsequent linear interpolation result in discontinuities in the respective objective functions. These discontinuities are not from any physical property of the structure or inherent to the problem, but are rather numerical artefacts arising from the automatic adjustment in arc length that makes analyses more computationally efficient and reliable. Note that even though the objective function contains discontinuities, the gradient or derivative of the objective function is computable for every design in the design space.

4.1 Benefits of automatic arc length adjustment

To make use of any gradient based optimization algorithm, the gradient of the objective function is required. If the gradient of the objective function is calculated at or near a discontinuity using a numerical approximation (such as forward differences), the returned gradient could be incorrect in both magnitude and direction [10]. To avoid these discontinuities two steps can be taken, namely, the analyst can use a small enough prescribed arc length such that the simulation never requires an arc length adjustment, or the analysis code can be adapted to always produce a solution coordinate at

the accumulated arc length value used to represent the target curve.

The first option, the small prescribed arc length, results in an increase in solution times. This can be demonstrated by solving the same symmetrical Deep Semi-Circular Arch for various prescribed arc lengths. Table 1 shows how for small prescribed arc length step sizes the solution times start to increase drastically when compared to a more appropriate arc length selection. Selecting a sufficiently small arc length step size is further complicated in that it cannot be determined *a priori* whether the arc length will be adjusted over all possible designs in the design space.

Table 1: Solution Times for Various Arc Length Step Sizes

Step Arc Length	**100**	**50**	**25**	**12.5**	**6.25**
Time (s)	26.2	42.5	53.67	89.61	163.75
Arc Adjustment	True	True	True	True	False

The second option of sampling the target and design curves at standard locations still requires the constructions of some interpolation functions for both the load and displacement as well as their respective gradients (whenever the original prescribed arc length step is too large to converge). As with any interpolation scheme, this will result in some loss of accuracy and can therefore still result in discontinuities.

Therefore, neither method of attempting to eliminate the discontinuities can entirely eliminate discontinuities from the objective function. In addition, the first proposed method can be computationally expensive. The complex-step method [10, 22] offers a means to compute sensitivities at a discontinuity but scales poorly with an increase in the number of design variables. This means that to make efficient use of gradient based algorithms it is necessary to implement an analytical sensitivity procedure. Appendices A.1 and A.2 detail the analytical sensitivity derivations and procedures.

5 Sensitivity Verification

To verify the proposed analytical sensitivity procedure, outlined in Appendixes A.1 and A.2, the same design problem as in Section 4, the Deep Semi-Circular Arch, is used. The original structure has a radius of 100mm and spans 270° (from -45° to 225°). The boundary conditions are asymmetric, with the left end pinned and the right end clamped. This asymmetry ensures that the structure will always initially deform to the left as the pinned boundary condition is less stiff than the clamped boundary condition. The geometry is parameterized using, 1, 2, 4 and 8 design variables as shown in Figure 14.

The chosen design variables describe the radius of the arch at equal angle increments from -45° to 225°, excluding the end points which remain at a radius of 100mm. The arch radius for any angle $\theta \in [-45°; 225°]$ is then described by a cubic spline.

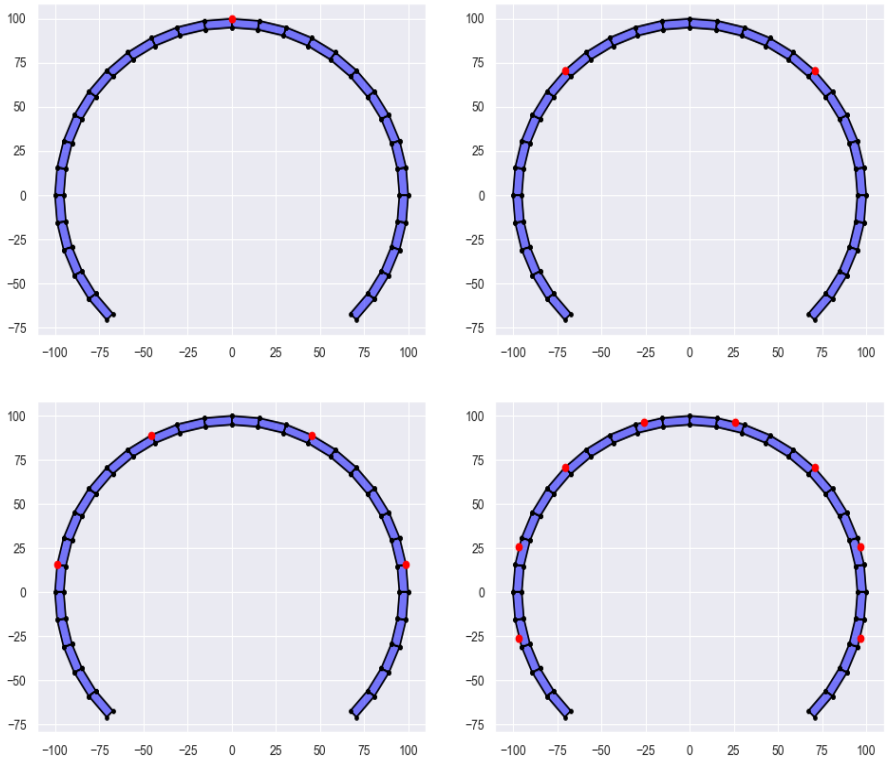


Figure 14: Shape parameter positions indicated in red for increasing dimensionality.

The effect of these shape variables on the load path can be calculated with a sensitivity study. This is done using a forward finite difference scheme as well as the proposed analytical sensitivity procedure. As the loads and displacements vary greatly in magnitude, the load paths are scaled using the minimum and maximum values of the target curves so that the load and displacement are weighted equally. The gradient of the objective function, Equation (9), is then expressed analytically as

$$\frac{dMSE}{dx} = -\frac{2}{N}\sum_{i}^{N}(\lambda_i^{\mathrm{T}} - \lambda_i^d)\frac{d\lambda_i^d}{dx} - \frac{2}{N}\sum_{i}^{N}(u_i^{\mathrm{T}} - u_i^d)\frac{du_i^d}{dx}, \tag{10}$$

where the load and displacement derivative terms, $\frac{d\lambda}{dx}$ and $\frac{du}{dx}$, for the analytical method are calculated from the proposed analytical sensitivity procedure.

Using the load path of a fully symmetrical arch shown in Figure 7 as the target curve, and

randomized designs for each parameterization with shape parameters between 80mm and 120mm, the partial derivatives w.r.t. each design variable can be compared between the two sensitivity methods. All structures are simulated with the same total arc length and prescribed arc length. Table 2 shows the calculated objective function gradient components using the two sensitivity methods for this example.

The results in Table 2 show that the numerical and analytical partial derivatives of the gradient vector for a random design agree up to the 6^{th} significant figure, when using a perturbation size of 10^{-6} in the forward finite difference scheme. Therefore the proposed analytical sensitivity procedure produces the correct results.

Table 2: Objective Function Gradient Vector: Forward Finite Difference vs. Analytical Procedure.

	Method											
# Var.	**Forward Finite Difference**				**Analytical Procedure**				**Ratio**			
1	-0.01003307	-	-	-	-0.01003310	-	-	-	0.999997	-	-	-
2	-0.00943673	-0.00449071	-	-	-0.00943646	-0.00449052	-	-	1.00002	1.00004	-	-
4	0.00415676	-0.02392256	0.02817192	-0.04849738	0.00415678	-0.02392294	0.02817125	-0.04849737	0.999995	0.999984	1.00002	1.000002

Another benefit of this analytical procedure is the increase in computational efficiency. Using finite difference schemes, multiple simulations per design variable are required to calculate the gradient of the objective function. This can become computationally expensive if multiple design variables are used or if the simulation itself is computationally expensive. The computational cost of the analytical method on the other hand is insensitive to the number of design variables. Table 3 shows the recorded solution times to compute the sensitivity results of this section.

Table 3: Sensitivity Calculation Times in seconds for Increasing Number of Variables

Number of Variables	**1**	**2**	**4**	**8**	**16**
Forward Difference	54.2	81	135	243	459
Analytical	48.1	48.2	48.1	48.6	48.5

6 Load Path Optimization

The performance of three different optimizers, that all make use of gradient information, are evaluated on the Deep Semi-Circular Arch load path optimization problem. The optimizers are the Sequential Least Squares Programming (SLSQP) method, the Gradient Only Sequential Spherical Approximation method (GOSSA) [10], and a Modified Subgradient method [10, 11]. The SLSQP method is a well known optimizer and has been used on a wide variety of numerical optimization problems [17]. The GOSSA and Modified Subgradient methods are gradient-only methods that have been shown by Wilke

[11], Wilke et al. [13], [14], [15] and Snyman and Wilke [10] to be reliable when solving problems that contain discontinuities.

The GOSSA algorithm estimates the Hessian of the objective function, making use of the spherical assumption, for each iteration. The efficiency of this algorithm should be comparable to the SLSQP algorithm, but it will not misinterpret discontinuities in the objective function as minima. The Modified Subgradient method is a simple steepest descent method with a chosen fixed step size (no line searches are performed) and the function minimizer along the search trajectory is *not* stored. This algorithm is also reliable in the presence of discontinuities but its performance is dependant on a hyper-parameter, meaning it can be computationally expensive when compared to the other algorithms.

The goal of the optimization problem is to find the values of the design variables that result in the same load path as the Deep Semi-Circular Arch, depicted in Figure 7. The load path of a fully symmetrical arch is chosen as the target curve as no matter the level of dimensionality, the fully symmetrical arch can be recovered exactly. This means the performance of the optimizers can be assessed easily as the global optimum is known (zero). The converged solutions of the SLSQP, GOSSA and Modified Subgradient methods will be referred to as $\mathbf{x}^*_{SLSQP}$, $\mathbf{x}^*_{GOSSA}$ and $\mathbf{x}^*_{MS}$ respectively in the sections that follow.

Inverse problems, such as the problem presented in this paper, are often ill-posed i.e. different designs (structures) have the same load-deflection path. In the case of this design problem this is inconsequential. The goal is to find *any* structure with the desired load-deflection path, not to find a *specific* structure with the target load-deflection path. Therefore, the uniqueness of the optimized solutions is not investigated.

6.1 Bifurcation of the Load-Deflection Path

The example design problem presented in this research, the Deep Semi-Circular Arch, can exhibit bifurcated load-deflection paths. Bifurcation refers to a loss of uniqueness in the solution to the non-linear FEM equilibrium equations. This loss of uniqueness typically presents itself as a "fork" in the equilibrium path, with there being multiple equilibrium positions of the structure for some value of the accumulated arc length parameter. The load-deflection curve then exhibits multiple branches. Figure 15 demonstrates the bifurcation present in the problem.

Although there are numerical strategies to consistently stay on the same branch, often referred

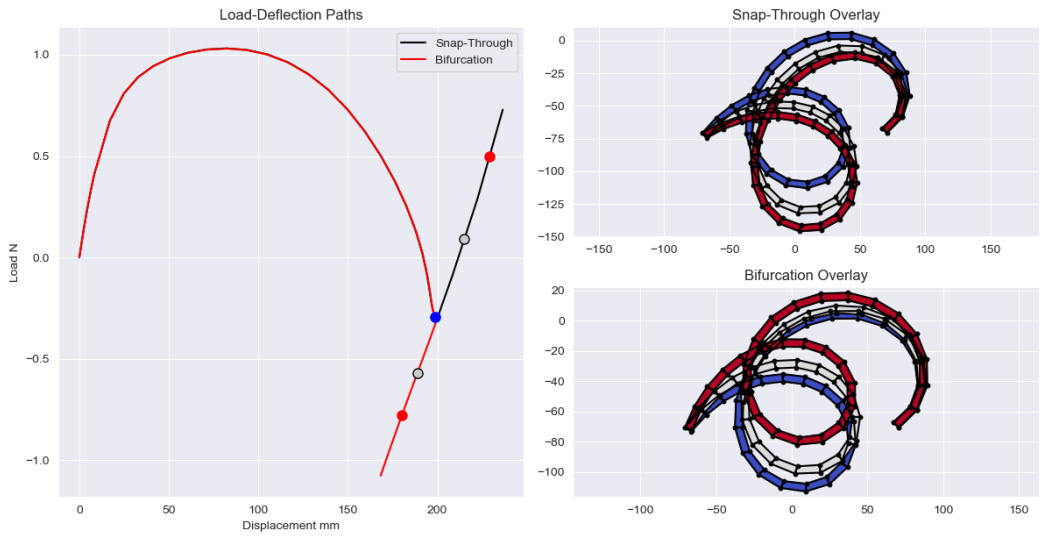


Figure 15: Demonstration of the type of bifurcation present in the example problem. The overlays show the "fork" equilibrium state in blue (identical for both figures) and the next two equilibrium states are shown in grey and red and their locations on the equilibrium paths.

to as branch switching algorithms [23, 24], the solver implemented in this research makes no effort to avoid these bifurcated load paths. Rather, the sudden increase in the objective function value if the solver finds equilibrium on a different branch, should result in the optimizers to moving away from areas in the design space where bifurcation is present. This can be demonstrated by selecting some initial starting points for the optimizers that describe structures that produce bifurcated load paths.

6.2 Two Shape Design Variables

The Two Variable Inverse Problem consists of two design variables at 45° and 135° that describe the radius at these locations (shown in Figure 14). To visualize the two objective functions described in Section 3, a grid of 75×75 samples between 80 mm and 120 mm is used. The resultant plots are shown in Figure 16. These 3D surface plots show the presence of similar discontinuities that were present in the one variable objective function. The global minimum is at the point $[100, 100]$.

The problem is solved with the selected optimizers using five randomly generated starting positions, two of which generate bifurcated load paths. The initial load path curves, and the target curve, are shown in Figure 17, while the returned optimized curves for the three methods are shown in Figure 18 for both objective functions. None of the returned optimal structures have bifurcated load-deflection curves, therefore the optimizers are able to bypass and avoid regions in the design space that result

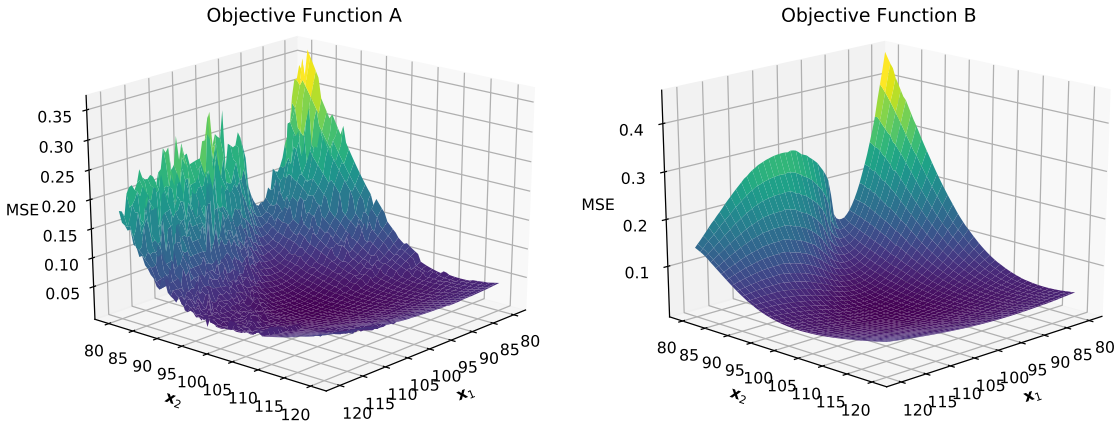


Figure 16: Objective functions of the Two Shape Design Variable Inverse Problem

in bifurcated load-deflection curves.

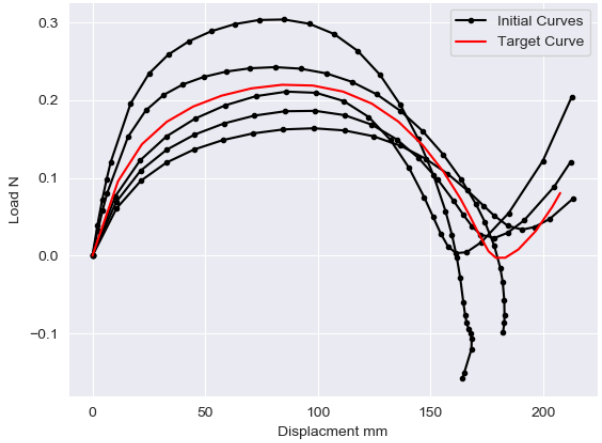


Figure 17: Initial Curves of the two variable optimization problem.

The SLSQP algorithm fails to find the optimal solution for three of the five initial designs for both objective functions although objective function **B** does offer a small improvement in results. The GOSSA and Modified Subgradient algorithms consistently find the global minimum for all the starting positions and both objective functions. To explain the cause of the occasional failure of the SLSQP algorithm, the optimization trajectories of the algorithms are presented in Figures 19 and 20.

These trajectories are overlaid with a quiver plot that shows the normalized gradient of the objective function (i.e. the arrows only show the direction and not the magnitude of the gradient vector).

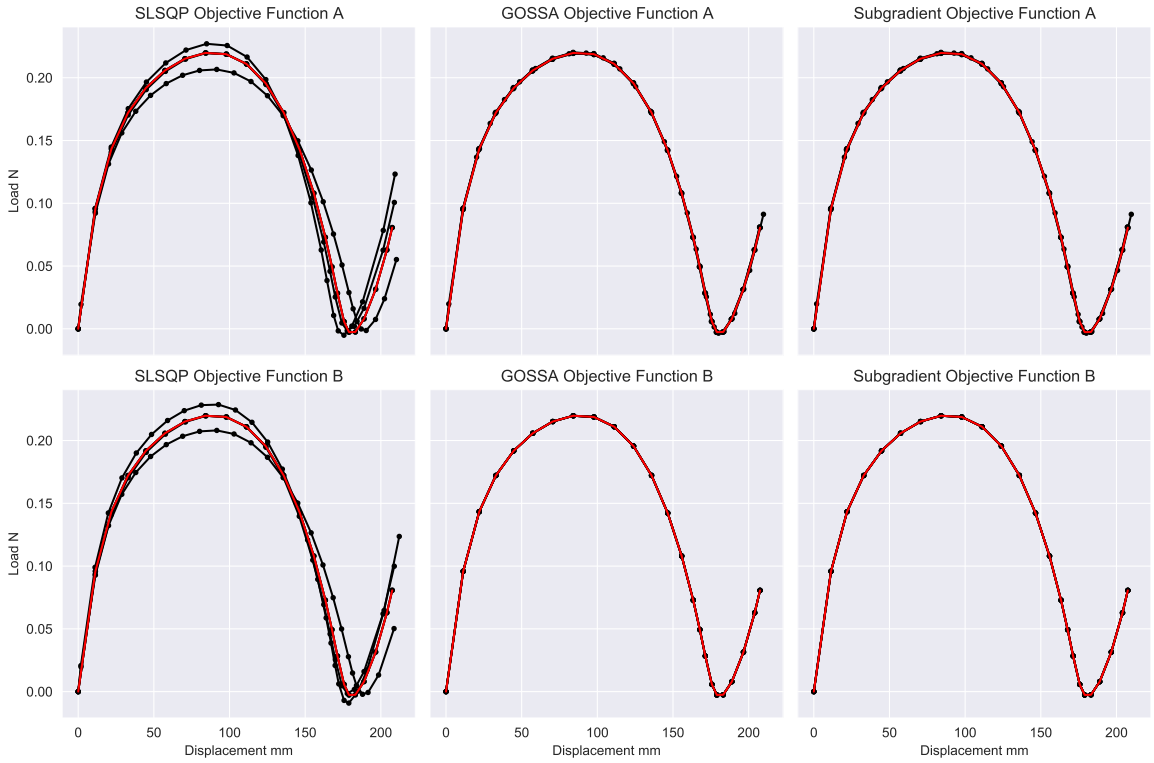


Figure 18: Optimized Curves for the Three Algorithms and both objective functions for the Two Variable Inverse Problem.

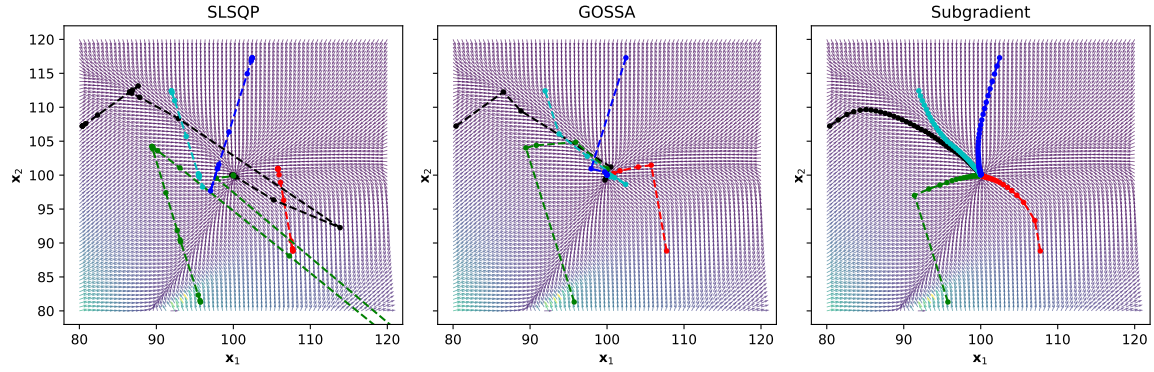


Figure 19: Optimization trajectories for objective function **A** on the Two Variable Inverse Problem.

An important aspect to note of these gradient fields is the similarity between the two objective functions. Objective function **A** has larger discontinuities when compared to objective function **B** (see Figure 16), but the resultant gradient fields for either objective function remain far more consis-

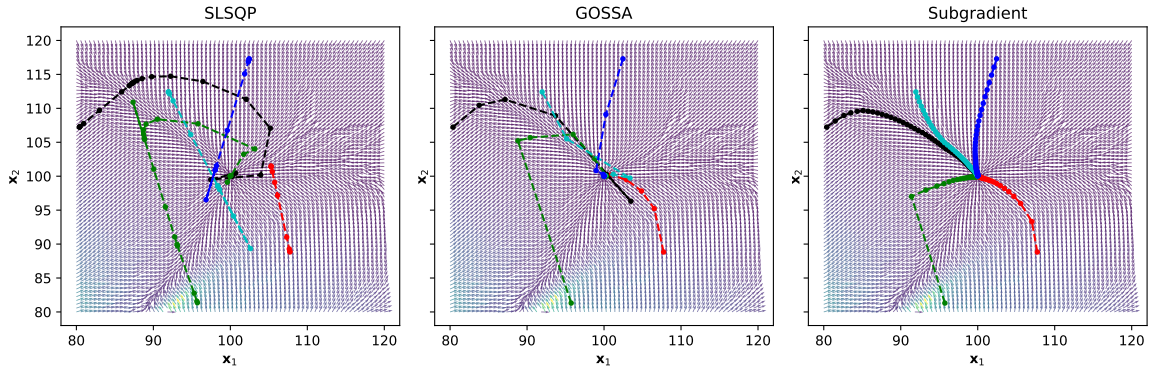


Figure 20: Optimization trajectories for objective function **B** on the Two Variable Inverse Problem.

tent. This means that the gradient information for the objective functions is far more reliable when compared to the discontinuous function value information.

The optimization trajectory plots show that the SLSQP and GOSSA algorithms both follow similar optimization paths, but the SLSQP method terminates early. This early termination is due to the discontinuities in the objective function caused by the adaptive stepping method of the ALC algorithm. This can be proven by completing a dense line search at these early termination points. Figure 21 presents an explicit line search

$$MSE(\alpha) = MSE(\mathbf{x}^*_{SLSQP} + \alpha \mathbf{u}^*_{SLSQP}), \tag{11}$$

along the SLSQP descent direction at convergence, $\mathbf{u}^*_{SLSQP}$, over a small range of α.

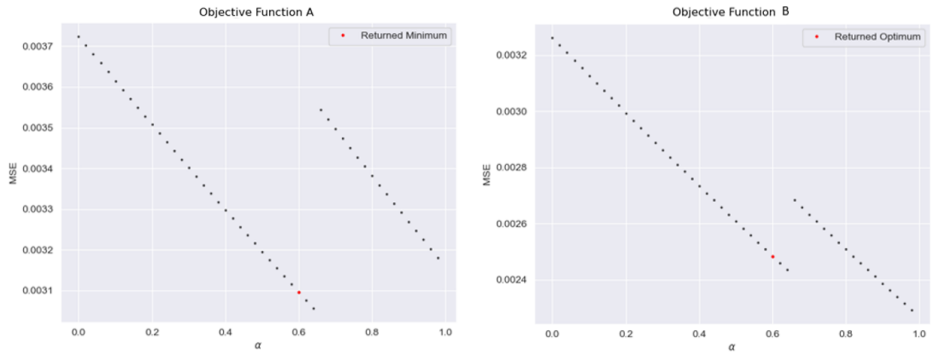


Figure 21: Line Searches at discontinuous local minimizers resulting in premature termination of SLSQP

SLSQP convergences to a discontinuity for both objective functions as sampling to the right of this

discontinuity results in an increase in the MSE. Consequently, SLSQP identifies the discontinuity as a local minimum and terminates. However, the directional derivative remains negative to the left and right of the discontinuity. Therefore, gradient-only optimizers proceed past the discontuinuity unhindered until the directional derivative changes sign from negative to positive, indicating a NN-GPP along a descent direction.

Tables 4 and 5 show the detailed results of the optimizers for this two dimensional problem, with the error term being the norm of the difference between the returned optimum vector and the known optimum. The norm of the gradient vector is two orders of magnitude larger at $\mathbf{x}^*_{SLSQP}$ than at $\mathbf{x}^*_{GOSSA}$ and $\mathbf{x}^*_{MS}$. This demonstrates that the gradient information at these discontinuities is not impacted as severely as the function value information. This means that by locating NN-GPPs, $\mathbf{x}^*_{\mathbf{NNGPP}}$, instead of minimizers, $\mathbf{x}^*$, the optimization algorithm is able to bypass the numerically induced discontinuities in the objective function. For this design problem, as the objective function is unimodal, the $\mathbf{x}^*_{\mathbf{N}}$ and $\mathbf{x}^*_{\mathbf{NNGPP}}$ optima agree exactly as the norm of the gradient vector at these positive gradient projection points are zero within a tolerance of 10^{-5}.

Table 4: Detailed objective function **A** Results for Two Variable Inverse Problem

Method	SLSQP				GOSSA				Modified Subgradient			
	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error
Start 1	14	23 / 14	2.62×10^{-3}	5.86	18	– / 27	2.38×10^{-5}	0.011	31	– / 31	2.38×10^{-5}	0.051
Start 2	20	22 / 20	3.24×10^{-5}	0.016	26	– / 33	4.13×10^{-6}	0.006	63	– / 63	2.20×10^{-5}	0.057
Start 3	20	25 / 20	8.53×10^{-5}	0.023	14	– / 19	2.53×10^{-5}	0.007	31	– / 31	2.17×10^{-5}	0.051
Start 4	18	30 / 18	3.13×10^{-3}	4.46	18	– / 26	9.43×10^{-6}	0.003	51	– / 51	2.48×10^{-5}	0.066
Start 5	10	11 / 10	4.66×10^{-3}	2.27	17	– / 27	4.67×10^{-6}	0.007	49	– / 49	2.19×10^{-5}	0.061

Table 5: Detailed objective function **B** Results for Two Variable Inverse Problem

Method	SLSQP				GOSSA				Modified Subgradient			
	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error
Start 1	10	20 / 10	1.42×10^{-3}	5.49	8	– / 10	2.77×10^{-6}	0.0012	20	– / 20	1.23×10^{-6}	0.002
Start 2	25	25 / 25	6.66×10^{-7}	0.0017	9	– / 14	2.80×10^{-6}	0.001	42	– / 42	2.41×10^{-6}	0.0016
Start 3	22	23 / 22	1.05×10^{-5}	0.028	8	– / 11	3.39×10^{-6}	0.0045	21	– / 21	1.89×10^{-6}	0.0048
Start 4	12	20 / 12	8.65×10^{-4}	2.3	7	– / 13	2.99×10^{-6}	0.0019	25	– / 25	2.85×10^{-6}	0.0032
Start 5	10	10 / 11	8.73×10^{-4}	2.17	6	– / 10	1.33×10^{-6}	0.0018	30	– / 30	3.45×10^{-6}	0.0024

6.3 Four and Eight Shape Design Variables

To further assess the ability of gradient-only optimizers, more complex problems are solved. Firstly, the problem is adapted to the Four Variable Inverse Problem (shown in Figure 14). As before, the target curve is that of a fully symmetrical arch. The initial curves are shown in Figure 22 while the

final optimized curves are shown in Figure 23.

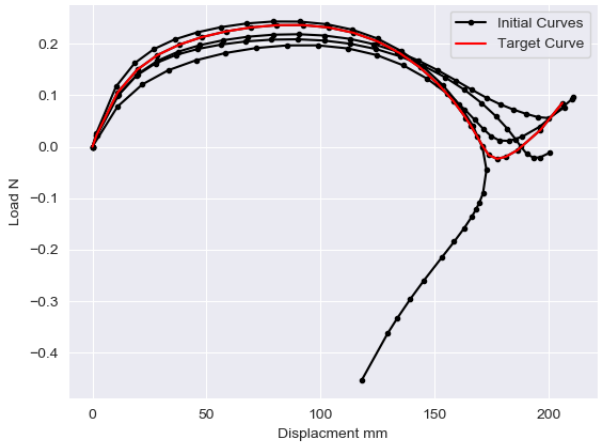


Figure 22: Initial Curves of Four Variable Optimization Problem

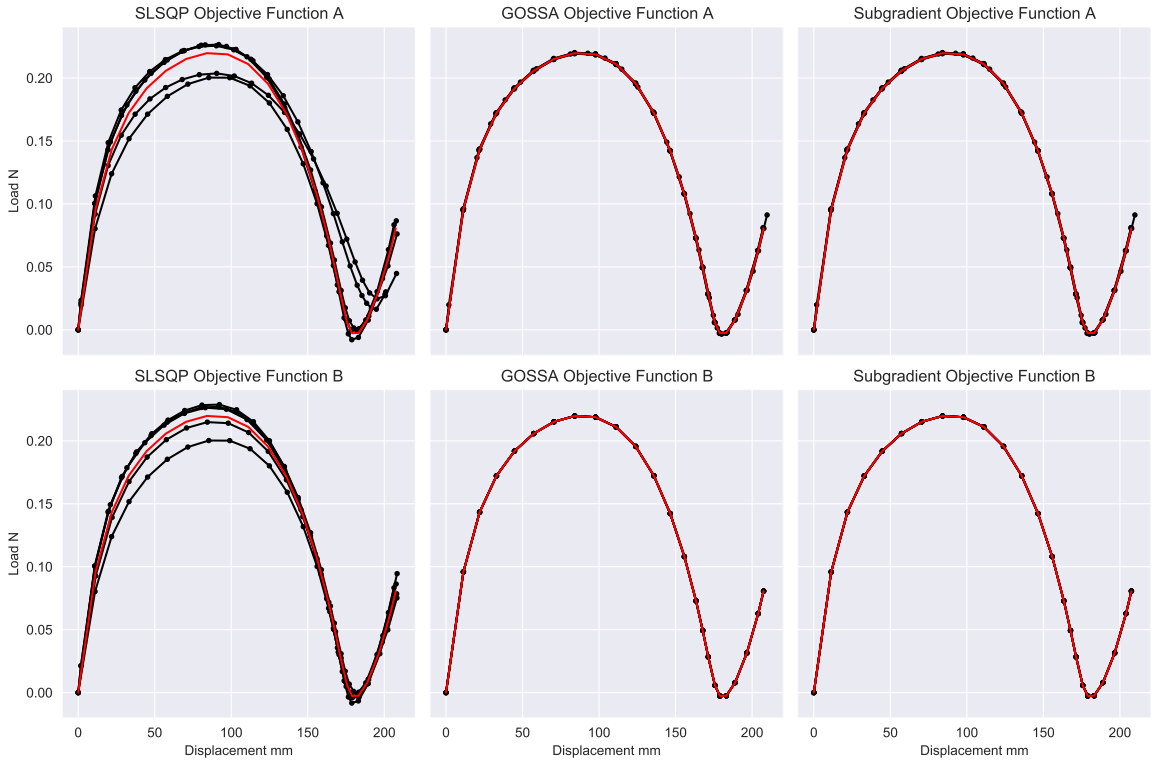


Figure 23: Optimized Curves for the Three Algorithms and both objective functions for the Four Variable Inverse Problem

The GOSSA and Modified Subgradient algorithms again find the global optimizer independent of the starting position in the domain and the implemented objective function. In contrast, the SLSQP algorithm struggled to find the optimum with none of the returned solutions fully tracing the target curve for either objective function, but objective function **B** again performs slightly better. Tables 6 and 7 again show detailed results of the three optimizers on the two objective functions. Although the GOSSA and Subgradient methods both returned the correct optimum curve, the increase in dimensionality begins to show the benefit in efficiency the GOSSA algorithm has compared to the Modified Subgradient method. The Modified Subgradient method took as many as three times the number of gradient computations to converge when compared to GOSSA. An indication that the SLSQP method again misinterpreted the discontinuities as minima is the large gradient norm at a discontinuous local minimizer when compared to the gradient norms at the GOSSA and Modified Subgradient optimizers.

Table 6: Detailed objective function **A** Results for Four Variable Inverse Problem

Method	SLSQP				GOSSA				Modified Subgradient			
	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error
Start 1	9	20 / 9	4.23×10^{-3}	8.17	22	– / 39	1.01×10^{-5}	1.34	38	– / 38	2.33×10^{-5}	1.29
Start 2	16	17 / 16	3.72×10^{-3}	7.86	34	– / 61	1.05×10^{-5}	2.58	189	– / 189	2.29×10^{-5}	4.95
Start 3	7	8 / 7	3.25×10^{-4}	9.53	20	– / 32	2.16×10^{-5}	3.15	94	– / 94	2.48×10^{-5}	3.04
Start 4	13	25 / 13	5.34×10^{-3}	8.84	20	– / 34	2.15×10^{-5}	0.785	78	– / 78	2.42×10^{-4}	0.76
Start 5	18	18 / 18	6.81×10^{-4}	4.51	26	– / 44	7.48×10^{-6}	3.01	71	– / 71	2.48×10^{-5}	3.02

Table 7: Detailed objective function **B** Results for the Four Variable Inverse Problem

Method	SLSQP				GOSSA				Modified Subgradient			
	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error
Start 1	15	15 / 15	2.13×10^{-4}	2.44	30	– / 30	1.25×10^{-5}	1.34	35	– / 35	1.36×10^{-5}	1.45
Start 2	18	18 / 18	3.14×10^{-4}	7.85	52	– / 52	2.34×10^{-5}	2.01	167	– / 167	1.86×10^{-5}	3.78
Start 3	8	8 / 8	8.24×10^{-4}	9.57	26	– / 26	2.14×10^{-5}	2.54	90	– / 90	2.52×10^{-5}	3.01
Start 4	18	18 / 18	4.43×10^{-4}	3.51	28	– / 28	1.65×10^{-5}	0.43	75	– / 75	1.12×10^{-5}	0.35
Start 5	16	17 / 16	4.41×10^{-4}	4.62	20	– / 20	2.89×10^{-6}	3.15	65	– / 65	2.23×10^{-6}	2.98

To verify that the SLSQP algorithm terminates at a discontinuous local minimizer along the descent direction, Figure 24 presents a graph of the objective function values returned by a line search conducted at one of the $\mathbf{x}^*_{SLSQP}$ solutions. As with the Two Variable Inverse Problem, the SLSQP method mistakes a discontinuity in the objective function for a local minimum, and terminates prematurely. As before, the GOSSA and Modified Subgradient algorithms are able to bypass these discontinuities and continue to the optimum solution by only making use of the gradient information.

The complexity of the problem is increased further to contain eight design variables (Figure 14).

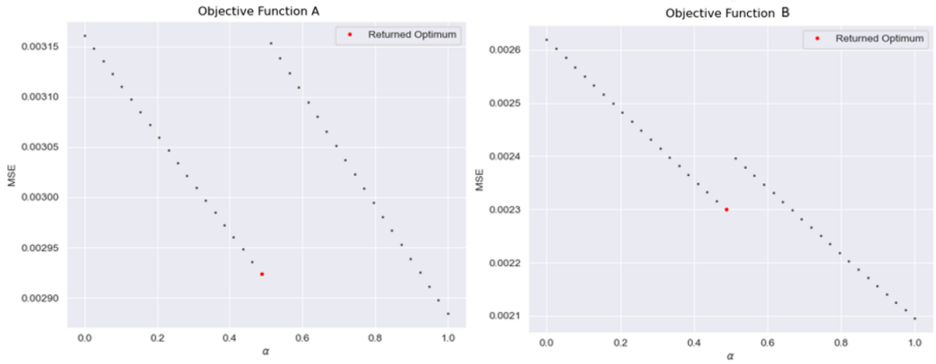


Figure 24: Line Searches at discontinuous local minimizers resulting in premature termination of SLSQP for the Four Variable Inverse Problem

The initial curves are shown in Figure 25, while the final optimized curves are shown in Figure 26 for the three optimizers.

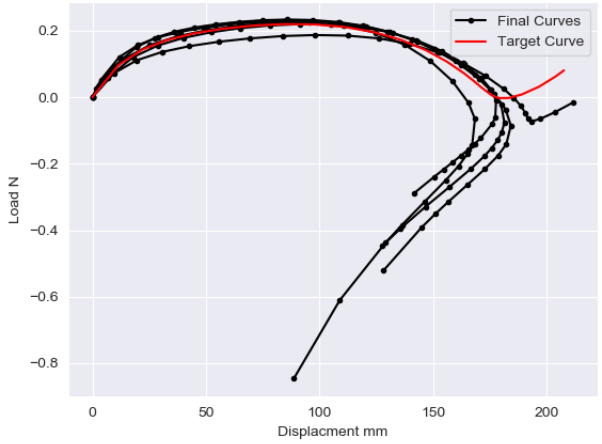


Figure 25: Initial Curves of Eight Variable Inverse Problem

The performance of the SLSQP algorithm, when using objective function **A**, deteriorates to such an extent that none of its optimized curves offer any reasonable similarity to the desired target curve. Objective function **B** performs better, with the majority of the solutions being similar to the target curve. The GOSSA and Modified Subgradient algorithms however remained reliable and still optimized fully to the desired target curve. Figure 27 illustrates some of the unsatisfactory final designs computed by the SLSQP algorithm.

Tables 8 and 9 show the detailed results for the implemented optimizers. The efficiency of the GOSSA algorithm is apparent at this level of dimensionality, as it consistently requires six to ten times

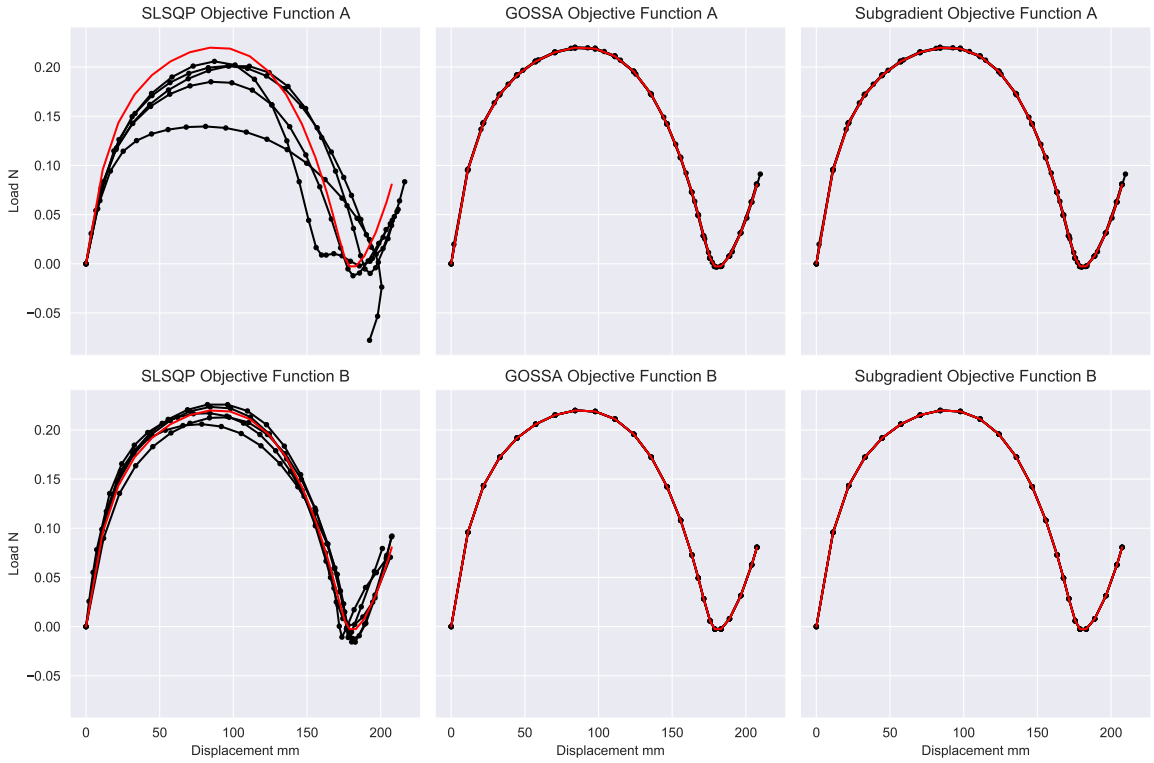


Figure 26: Optimized Curves for the Three Algorithms and both objective functions for the Eight Variable Inverse Problem

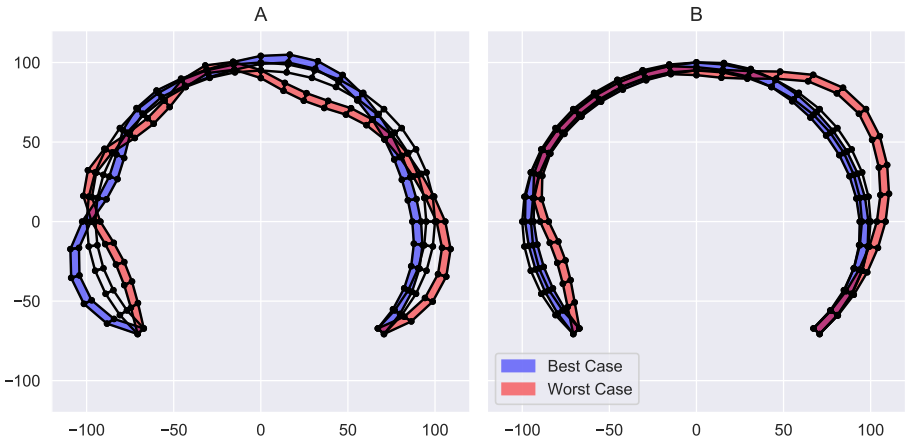


Figure 27: Optimized Shapes returned by the SLSQP Algorithm for the Eight Variable Inverse Problem Overlaid with the Global Optimum. The best and worst results are indicated in blue and red respectively, using Objective Function **A** and Objective Function **B**.

fewer simulations than the Modified Subgradient method to converge. Again, by simply inspecting the gradient norm at the returned optimums, it becomes apparent that the SLSQP algorithm struggled to bypass the discontinuities in the objective function.

Table 8: Detailed objective function **A** Results for Eight Variable Inverse Problem

Method	SLSQP				GOSSA				Modified Subgradient			
	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	Iterations	Evaluations (Func/Jac)	Gradient Norm	Error	Iterations	Evaluations (Func/Jac)	Gradient Norm	Error
Start 1	23	42 / 23	4.10×10^{-2}	25.93	95	– / 150	2.25×10^{-5}	10.62	359	– / 359	1.24×10^{-5}	4.61
Start 2	16	19 / 16	1.62×10^{-2}	20.81	53	– / 92	1.07×10^{-5}	12.72	279	– / 279	4.99×10^{-5}	13.61
Start 3	11	22 / 11	3.64×10^{-2}	47.53	16	– / 29	5.61×10^{-5}	18.91	728	– / 728	5.44×10^{-5}	17.31
Start 4	18	30 / 18	2.00×10^{-2}	16.70	37	– / 59	2.11×10^{-5}	3.93	221	– / 221	4.99×10^{-5}	3.89
Start 5	16	46 / 16	2.81×10^{-2}	13.19	64	– / 111	1.56×10^{-5}	4.96	241	– / 241	4.99×10^{-5}	5.16

Table 9: Detailed objective function **B** Results for Eight Variable Inverse Problem

Method	SLSQP				GOSSA				Modified Subgradient			
	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error	# Iter.	Evaluations (Func/Jac)	Gradient Norm	Error
Start 1	41	51 / 41	4.31×10^{-4}	22.35	80	– / 80	1.89×10^{-5}	9.86	301	– / 301	1.45×10^{-5}	6.81
Start 2	35	36 / 35	4.51×10^{-4}	14.21	45	– / 45	2.56×10^{-6}	13.21	254	– / 254	2.71×10^{-6}	12.34
Start 3	27	35 / 27	3.56×10^{-4}	28.98	20	– / 20	3.54×10^{-5}	17.41	689	– / 689	3.78×10^{-5}	16.31
Start 4	34	38 / 34	2.21×10^{-4}	5.06	30	– / 30	1.34×10^{-5}	3.54	214	– / 214	1.93×10^{-5}	3.84
Start 5	20	21 / 20	3.48×10^{-4}	10.11	54	– / 54	2.84×10^{-5}	4.84	231	– / 231	2.46×10^{-5}	4.96

Figure 28 again shows a line search at a local minimizer $\mathbf{x}^*_{SLSQP}$ along the descent direction for SLSQP. As the complexity and dimensionality of the design problem increased, the size and number of the discontinuities increased to the point where the SLSQP algorithm can no longer offer reasonable performance. The gradient-only optimizers isolating NN-GPPs remained reliable as the complexity of the problem increased. The number or size of the discontinuities did not affect the quality of the results.

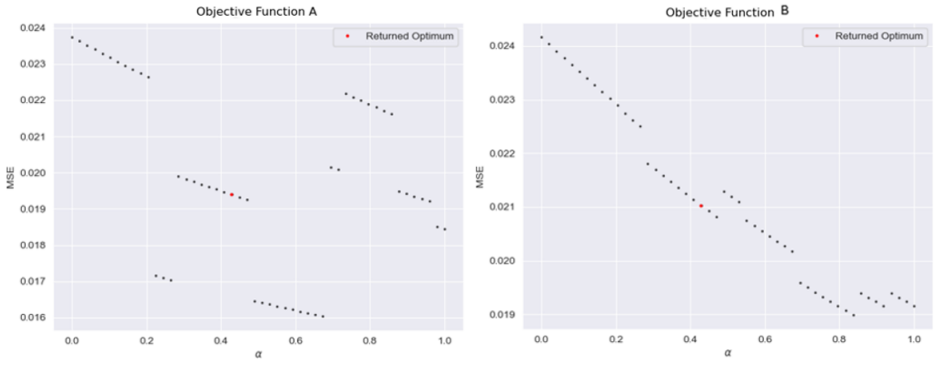


Figure 28: Line Searches for Eight Variable Inverse Problem at discontinuous local minimizers resulting in premature termination of SLSQP

.

7 Conclusion

The results in this paper show that an optimization problem that requires the ALC algorithm when computing the objective function, will exhibit discontinuities in the objective function if automatic arc length adjustment takes place during the simulation. Instead of deactivating automatic arc length control, it is retained in this work since it ensures that the algorithm is computationally efficient, and that the load-deflection path is computable for all designs. However, the resulting discontinuities are purely numerical artefacts, and the chosen optimization algorithms should ignore them when searching for the optimal solutions.

The presence of discontinuities also implicitly limits the use of numerical approximations (such as forward finite difference) to compute the gradient of the objective function, as such methods can return incorrect results if the calculation takes place across one of these discontinuities. Although the complex-step method might offer an alternative to compute sensitivities, we opt for an analytical sensitivity procedure to calculate the gradient of the objective function that remains efficient as the dimensionality increases.

As SLSQP aims to locate function minimizers $\mathbf{x}^*$, it is prone to terminate prematurely at discontinuities in the objective function, that presents as local minimizers along the search directions. This renders classical gradient based algorithms vulnerable. In turn, gradient-only optimizers that aim to isolate NN-GPPs are able to reliably locate the global optimizer, making it computationally efficient for high-dimensional problems. In particular, GOSSA proved more efficient than the Modified Subgradient method, and also does not require any hyper-parameter tuning.

The results in this paper provide compelling evidence that it is possible to design snap-through structures that match the desired load-deflection path without limiting the complexity of both the desired and simulated load-deflection paths. Essential features of the proposed approach are automatic arc length adjustment to ensure efficient analysis, combined with gradient-only optimization algorithms that reliably locate NN-GPPs in the design space.

References

[1] A. Bhattacharyya, C. Conlan-Smith, and K. A. James, "Design of a Bi-stable Airfoil with Tailored Snap-through Response Using Topology Optimization," *CAD Computer Aided Design*, vol. 108, pp. 42–55, 2019. [Online]. Available: https://doi.org/10.1016/j.cad.2018.11.001

[2] H. Deng, L. Cheng, X. Liang, D. Hayduke, and A. C. To, "Topology optimization for energy dissipation design of lattice structures through snap-through behavior," *Computer Methods in Applied Mechanics and Engineering*, vol. 358, p. 112641, 2020. [Online]. Available: https://doi.org/10.1016/j.cma.2019.112641

[3] T. Sekimoto and H. Noguchi, "Homologous Topology Optimization in Large Displacement and Buckling Problems," *JSME International Journal*, vol. 44, no. 4, pp. 616 – 622, 2001.

[4] T. E. Bruns, O. Sigmund, and D. A. Tortorelli, "Numerical methods for the topology optimization of structures that exhibit snap-through," *International Journal for Numerical Methods in Engineering*, vol. 55, no. 10, pp. 1215–1237, 2002.

[5] T. E. Bruns and O. Sigmund, "Toward the topology design of mechanisms that exhibit snap-through behavior," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 36-38, pp. 3973–4000, 2004.

[6] S. Kok and D. N. Wilke, "Optimizing snap-through structures by using gradient-only algorithms," in *11th World Congress on Structural and Multidisciplinary Optimisation, Sydney, Australia*, 2015, pp. 7–12.

[7] D. DaDeppo and E. Schmidt, "Instability of clamped-hinged circular arches subjected to a point load," *Transactions of the American Society of Mechanical Engineers, Journal of Applied Mechanics*, pp. 894—896, Dec. 1975.

[8] I. Leahu-Aluas and F. Abed-Meraim, "A proposed set of popular limit-point buckling benchmark problems," *Structural Engineering and Mechanics*, vol. 38, no. 6, pp. 767–802, 2011.

[9] E. Riks, "The application of newton's method to the problem of elastic stability," *Journal of Applied Mechanics, Transactions ASME*, vol. 39, no. 4, pp. 1060–1065, 1972.

[10] J. A. Snyman and D. N. Wilke, *Practical Mathematical Optimization*, 2nd ed. Springer, 2018.

[11] D. Wilke, "Modified subgradient methods for remeshing based structural shape optimization," in *Proceedings of the 13th International Conference on Civil, Structural and Environmental Engineering Computing*, 2011.

[12] ——, "Structural shape optimization using shor's r-algorithm," in *Third International Conference on Engineering Optimization*, 2012.

[13] D. Wilke, J. Snyman, S. Kok, and A. Groenwold, "Gradient-only approaches to avoid spurious local minima in unconstrained optimization," *Optimization and Engineering*, vol. 14, 06 2011.

[14] D. Wilke, "Approaches to accommodate remeshing in shape optimization," Ph.D. dissertation, University of Pretoria, 08 2010.

[15] D. Wilke, S. Kok, and A. Groenwold, "Relaxed error control in shape optimization that utilizes remeshing," *International Journal for Numerical Methods in Engineering*, vol. 94, pp. 273–289, 04 2013.

[16] D. N. Wilke, Kok, Schalk, and A. A. Groenwold, "The application of gradient-only optimization methods for problems discretized using non-constant methods," *Structural and Multidisciplinary Optimization*, vol. 40, pp. 433–451, 2010.

[17] D. Kraft, "A software package for sequential quadratic programming," DLR German Aerospace Center – Institute for Flight Mechanics, Koln, Germany., Tech. Rep., 1988.

[18] M. Ritto-Corrêa and D. Camotim, "On the arc-length and other quadratic control methods: Established, less known and new implementation procedures," *Computers and Structures*, vol. 86, no. 11-12, pp. 1353–1368, 2008.

[19] M. A. Crisfield, "A fast incremental/iterative solution procedure that handles "snap-through"," *Computers & Structures*, vol. 13, no. 1, pp. 55–62, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0045794981901085

[20] M. Bashir-ahmed and S. U. Xiao-zu, "Arc-length technique for nonlinear finite element analysis," *Journal of Zhejiang University SCIENCE*, vol. 5, no. 5, pp. 618–628, 2004.

[21] T. H. Pian and K. Sumihara, "Rational approach for assumed stress finite elements," *International Journal for Numerical Methods in Engineering*, vol. 20, no. 9, pp. 1685–1695, 1984.

[22] D. N. Wilke and S. Kok, "Numerical sensitivity computation for discontinuous gradient-only optimization problems using the complex-step method," *Proceedings of the Tenth World Congress on Computational Mechanics (WCCM 2012)*, vol. 1, pp. 3665–3676, 2014.

[23] H. B. Keller, "Constructive methods for bifurcation and nonlinear eigenvalue problems," in *Computing Methods in Applied Sciences and Engineering, 1977, I*, R. Glowinski, J. L. Lions, and I. Laboria, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 241–251.

[24] W. C. Rheinboldt, "Numerical methods for a class of finite dimensional bifurcation problems," *SIAM Journal on Numerical Analysis*, vol. 15, no. 1, pp. 1–11, 1978. [Online]. Available: https://doi.org/10.1137/0715001

[25] Y. S. Ryu, M. Haririan, C. C. Wu, and J. S. Arora, "Structural design sensitivity analysis of nonlinear response," *Computers and Structures*, vol. 21, no. 1-2, pp. 245–255, 1985.

[26] N. Olhoff and E. Lund, "Finite Element Based Engineering Design Sensitivity Analysis and Optimization," Ph.D. dissertation, Aalborg University, 1995.

[27] T. Hisada, "Recent Progress in Nonlinear FEM-Based Sensitivity Analysis," *JSME International Journal*, vol. 38, no. 3, pp. 430 – 433, 1995.

[28] J. Parente and L. E. Vaz, "On evaluation of shape sensitivities of non-linear critical loads," *International Journal for Numerical Methods in Engineering*, vol. 56, no. 6, pp. 809–846, 2003.

A Sensitivity Derivations

A.1 Arc Length Control Sensitivity

In this section analytical shape sensitivities [25–27] are computed that includes the ALC procedure for multiple limit points. To the best of our knowledge, analytical shape sensitivities that includes the ACL procedure is limited to the first limit point [27, 28], without offering detailed guidelines on how to proceed for multiple limit points.

The governing residual equation, with its dependencies, of a non-linear FEM problem at equilibrium is expressed as

$$\mathbf{R}(\mathbf{u}(\lambda(\mathbf{x}), \mathbf{x}), \lambda(\mathbf{x}), \mathbf{x}) = \mathbf{0}, \tag{12}$$

where $\mathbf{u}$, λ and $\mathbf{x}$ denote the nodal displacement vector, the load parameter, and the design vector respectively. The gradient with respect to the design variable vector $\mathbf{x}$ is then expressed as

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \lambda} \frac{\partial \lambda}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \lambda} \frac{\partial \lambda}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{x}} = \mathbf{0}. \tag{13}$$

The terms $\frac{\partial \mathbf{R}}{\partial \mathbf{u}}$ and $\frac{\partial \mathbf{R}}{\partial \lambda}$ are simply the tangent stiffness matrix $\mathbf{K}_T$ and the load vector $\mathbf{F}$ respectively. Therefore, Equation (13) can be re-written as

$$\mathbf{K}_T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \left(\mathbf{K}_T \frac{\partial \mathbf{u}}{\partial \lambda} + \mathbf{F} \right) \frac{\partial \lambda}{\partial \mathbf{x}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}}. \tag{14}$$

Expressing the derivative of Equation (12) with respect to the load parameter results in the equation

$$\mathbf{K}_T \frac{\partial \mathbf{u}}{\partial \lambda} + \mathbf{F} = \mathbf{0}. \tag{15}$$

Substituting Equation (15) into Equation (14) results in the system of equations

$$\mathbf{K}_T \frac{\partial \mathbf{u}}{\partial \mathbf{x}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}}. \tag{16}$$

This system of equations is then solved to find the displacement gradient term $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$. As the design variable term is a vector of values the resultant displacement derivative term is a $N \times M$ matrix where N is the degrees of freedom in the mesh and M is the number of shape parameters in the design problem. The derivative of the residual vector with respect to the design variable vector, $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$,

is dependant on the implemented solution scheme and the element type used in the mesh. This term is discussed and derived in Section A.2.

In the case of the ALC algorithm, the displacement vector $\mathbf{u}$ and the load parameter λ both feature in the Arc Length constraint equation

$$\blacktriangle\mathbf{u}(\blacktriangle\lambda(\mathbf{x}),\mathbf{x})^{\mathrm{T}}\blacktriangle\mathbf{u}(\blacktriangle\lambda(\mathbf{x}),\mathbf{x})+\psi^2\blacktriangle\lambda(\mathbf{x})^2=L^2. \tag{17}$$

Take note of the difference between the $\mathbf{u}$, λ and $\blacktriangle\mathbf{u}$, $\blacktriangle\lambda$ terms in the Arc Length constraint equation. The Arc Length constraint equation is satisfied exactly at each iteration while attempting to find the equilibrium solution for a particular load step in the solution process [18]. The variables $\blacktriangle\mathbf{u}$ and $\blacktriangle\lambda$ denote the *total incremental* change of the nodal displacements and load parameter from the beginning to the end of a load step (i.e. adding all the iterative changes, $\Delta\mathbf{u}$ and $\Delta\lambda$, for a particular load step). The derivative of the Arc Length constraint equation is then expressed as

$$2\blacktriangle\mathbf{u}^{\mathrm{T}}\left(\frac{\partial\blacktriangle\mathbf{u}}{\partial\blacktriangle\lambda}\frac{\partial\blacktriangle\lambda}{\partial\mathbf{x}}+\frac{\partial\blacktriangle\mathbf{u}}{\partial\mathbf{x}}\right)+2\psi^2\blacktriangle\lambda\frac{\partial\blacktriangle\lambda}{\partial\mathbf{x}}=0, \tag{18}$$

which can be simplified into

$$\blacktriangle\mathbf{u}^{\mathrm{T}}\frac{\partial\blacktriangle\mathbf{u}}{\partial\mathbf{x}}+\left(\blacktriangle\mathbf{u}^{\mathrm{T}}\frac{\partial\blacktriangle\mathbf{u}}{\partial\blacktriangle\lambda}+\psi^2\blacktriangle\lambda\right)\frac{\partial\blacktriangle\lambda}{\partial\mathbf{x}}=0. \tag{19}$$

What remains is to convert the *total incremental* terms in Equation (19) to the *total* terms in Equation (13). This can be done by expressing the *total* terms as the summation of all the previous *total incremental* terms,

$$\mathbf{u}=\sum_{i}^{N}\blacktriangle\mathbf{u}_i, \tag{20}$$

$$\lambda=\sum_{i}^{N}\blacktriangle\lambda_i, \tag{21}$$

and then rearranging the equations such that current *incremental* terms at solution step N, is the difference between the total terms and the summation of all the previous *incremental* terms, meaning from the summation up to $N-1$,

$$\blacktriangle\mathbf{u}_N=\mathbf{u}-\sum_{i}^{N-1}\blacktriangle\mathbf{u}_i, \tag{22}$$

$$\blacktriangle\lambda_N = \lambda - \sum_i^{N-1} \blacktriangle\lambda_i. \tag{23}$$

Computing the gradients of Equation (22) and (23) w.r.t. the design variables results in

$$\frac{\partial \blacktriangle\mathbf{u}_N}{\partial \mathbf{x}} = \frac{\partial \mathbf{u}}{\partial \lambda}\frac{\partial \lambda}{\partial \mathbf{x}} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}} - \sum_i^{N-1} \frac{d\blacktriangle\mathbf{u}_i}{d\mathbf{x}}, \tag{24}$$

$$\frac{\partial \blacktriangle\lambda_N}{\partial \mathbf{x}} = \frac{\partial \lambda}{\partial \mathbf{x}} - \sum_i^{N-1} \frac{d\blacktriangle\lambda_i}{d\mathbf{x}}. \tag{25}$$

These equations are then substituted into Equation (19),

$$\begin{aligned} \blacktriangle\mathbf{u}^{\mathrm{T}} \left(\frac{\partial \mathbf{u}}{\partial \lambda}\frac{\partial \lambda}{\partial \mathbf{x}} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}} - \sum_i^{N-1} \frac{d\blacktriangle\mathbf{u}_i}{d\mathbf{x}} \right) \\ + \left(\blacktriangle\mathbf{u}^{\mathrm{T}} \frac{\partial \blacktriangle\mathbf{u}}{\partial \blacktriangle\lambda} + \psi^2 \blacktriangle\lambda \right) \left(\frac{\partial \lambda}{\partial \mathbf{x}} - \sum_i^{N-1} \frac{d\blacktriangle\lambda_i}{d\mathbf{x}} \right) = 0, \end{aligned} \tag{26}$$

and then rearranged into

$$\begin{aligned} \blacktriangle\mathbf{u}^{\mathrm{T}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \left(\blacktriangle\mathbf{u}^{\mathrm{T}} \frac{\partial \mathbf{u}}{\partial \lambda} + \psi^2 \blacktriangle\lambda \right) \frac{\partial \lambda}{\partial \mathbf{x}} = \\ \blacktriangle\mathbf{u}^{\mathrm{T}} \sum_i^{N-1} \frac{d\blacktriangle\mathbf{u}_i}{d\mathbf{x}} + \left(\blacktriangle\mathbf{u}^{\mathrm{T}} \frac{\partial \mathbf{u}}{\partial \lambda} + \psi^2 \blacktriangle\lambda \right) \sum_i^{N-1} \frac{d\blacktriangle\lambda_i}{d\mathbf{x}}. \end{aligned} \tag{27}$$

This equation can then be used to solve for the load parameter derivative $\frac{\partial \lambda}{\partial \mathbf{x}}$ as all the other terms in the equation are known. The incremental terms, $\blacktriangle\mathbf{u}$ and $\blacktriangle\lambda$, are known as the sensitivity calculation occurs at the end of a solution step, and the gradient terms $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{u}}{\partial \lambda}$ are calculated from Equations (16) and (15) respectively. For the first solution increment the terms $\sum_i^{N-1} \frac{d\blacktriangle\mathbf{u}}{d\mathbf{x}}$ and $\sum_i^{N-1} \frac{d\blacktriangle\lambda}{d\mathbf{x}}$ are zero.

A key difference that the derivations demonstrate between typical sensitivity analysis, where load or displacement control is used, and the sensitivity with the ALC algorithm is that the sensitivity calculation is a function of the solution variables at previous iterations within the current load step, where typically the sensitivity calculation is independent of the solution variables at intermediate iterations.

A.2 Residual Sensitivity

The residual derivative, $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$, can be calculated using the chain rule

$$\frac{\partial \mathbf{R}}{\partial \mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathcal{X}} \frac{\partial \mathcal{X}}{\partial \mathbf{x}}, \tag{28}$$

where $\mathcal{X}$ denotes the global coordinates of the nodes in the mesh. Therefore the term $\frac{\partial \mathcal{X}}{\partial \mathbf{x}}$ quantifies the effect the design variables have on the nodal coordinates of the mesh. A forward finite difference scheme, where the design variables are altered by some small value and the structure re-meshed, is used in this paper to compute this term.

Since structures that exhibit highly non-linear behaviour, such as snap-through, typically exhibit this behaviour when loaded in bending, the 4-noded assumed stress element [21] is used for the simulations in this research. This element type is exact in bending for linear elasticity, and remains highly accurate in bending for non-linear elasticity. The internal residual and tangent stiffness matrix of the assumed stress element are expressed as

$$\mathbf{R}^{\text{int}} = \sum_e (t_e[\mathbf{G}_e][\beta_e] - \mathbf{F}_e), \tag{29}$$

$$[\mathbf{K}] = \sum_e t_e([\mathbf{L}_e] + [\mathbf{G}_e][\mathbf{H}_e]^{-1}[\mathbf{G}_e]^{\mathrm{T}}), \tag{30}$$

where $\mathbf{F}_e$ is the nodal load vector, t_e is the thickness of the element and the remaining terms, $[\mathbf{G}_e]$, $[\beta_e]$, $[\mathbf{L}_e]$, and $[\mathbf{H}_e]$ are typically calculated using Gauss quadrature. These terms are calculated from

$$[\mathbf{G}_e] = \sum_{GP} [\mathbf{B}]^{\mathrm{T}}[\mathcal{F}][\mathbf{B}]\det([\mathbf{J}])W_{GP}, \tag{31}$$

$$[\beta_{\mathbf{e}}] = [\mathbf{H}_e]^{-1}[\mathbf{M}_e], \tag{32}$$

$$[\mathbf{M}_e] = \sum_{GP} [\mathbf{P}]^{\mathrm{T}}[\mathbf{E}]\det([\mathbf{J}])W_{GP} \tag{33}$$

$$[\mathbf{L}_e] = \sum_{GP} [\mathbf{B}]^{\mathrm{T}}[\mathbf{S}][\mathbf{B}]\det([\mathbf{J}])W_{GP}, \tag{34}$$

$$[\mathbf{H}_e] = \sum_{GP} [\mathbf{P}]^{\mathrm{T}} [\mathbf{C}]^{-1} [\mathbf{P}] \det([\mathbf{J}]) W_{GP}, \tag{35}$$

where the terms [**J**], [**B**], [**P**], [**S**], [**E**], and [**C**] are the Jacobian matrix, strain-displacement matrix, stress interpolation matrix, the second Piola-Kirchoff stress matrix, the Green-Lagrange strain, and the material constant matrix respectively. The Gauss quadrature weights are indicated by W_{GP}.

The derivatives of the internal residual vector and the tangent stiffness matrix with respect to the nodal coordinates are

$$\frac{d\mathbf{R}^{\mathrm{int}}}{d\mathcal{X}} = \sum_e t_e \left(\frac{d[\mathbf{G}_e]}{d\mathcal{X}} [\beta_e] + [\mathbf{G}_e] \frac{d[\beta_e]}{d\mathcal{X}} \right), \tag{36}$$

$$\begin{aligned} \frac{d[\mathbf{K}]}{d\mathcal{X}} = \sum_e t_e \Bigg(& \frac{d[\mathbf{L}_e]}{d\mathcal{X}} + \frac{d[\mathbf{G}_e]}{d\mathcal{X}} [\mathbf{H}_e]^{-1} [\mathbf{G}_e]^{\mathrm{T}} \\ & + [\mathbf{G}_e] \frac{d[\mathbf{H}_e]^{-1}}{d\mathcal{X}} [\mathbf{G}_e] + [\mathbf{G}_e] [\mathbf{H}_e]^{-1} \frac{d[\mathbf{G}_e]}{d\mathcal{X}} \Bigg). \end{aligned} \tag{37}$$

Therefore, the required derivatives of Equations (31) – (35) are expressed as

$$\begin{aligned} \frac{d[\mathbf{G}_e]}{d\mathcal{X}} = \sum_{GP} \Bigg(& \frac{d[\mathbf{B}^{\mathrm{T}}]}{d\mathcal{X}} [\mathcal{F}] [\mathbf{B}] \det([\mathbf{J}]) \\ & + [\mathbf{B}]^{\mathrm{T}} \frac{d[\mathcal{F}]}{d[\mathcal{X}]} [\mathbf{B}] \det([\mathbf{J}]) \\ & + [\mathbf{B}] [\mathcal{F}] \frac{d[\mathbf{B}]}{d[\mathcal{X}]} \det([\mathbf{J}]) + [\mathbf{B}]^{\mathrm{T}} [\mathcal{F}] [\mathbf{B}] \frac{d \det([\mathbf{J}])}{d\mathcal{X}} \Bigg) W_{GP}, \end{aligned} \tag{38}$$

$$\frac{d[\beta_{\mathbf{e}}]}{d\mathcal{X}} = \frac{d[\mathbf{H}_e]^{-1}}{d\mathcal{X}} [\mathbf{M}_e] + [\mathbf{H}_e]^{-1} \frac{d[\mathbf{M}_e]}{d\mathcal{X}}, \tag{39}$$

$$\begin{aligned} \frac{d[\mathbf{M}_e]}{d\mathcal{X}} = \sum_{GP} \Bigg(& \frac{d[\mathbf{P}]^{\mathrm{T}}}{d\mathcal{X}} [\mathbf{E}] \det([\mathbf{J}]) \\ & + [\mathbf{P}]^{\mathrm{T}} \frac{d[\mathbf{E}]}{d\mathcal{X}} [\mathbf{P}] \det([\mathbf{J}]) \\ & + [\mathbf{P}]^{\mathrm{T}} [\mathbf{E}] \frac{d \det([\mathbf{J}])}{d\mathcal{X}} \Bigg) W_{GP}, \end{aligned} \tag{40}$$

$$\frac{d[\mathbf{L}_e]}{d\mathcal{X}} = \sum_{GP} \Bigg([\frac{d\mathbf{B}]^{\mathrm{T}}}{d\mathcal{X}}[\mathbf{S}][\mathbf{B}]\det([\mathbf{J}]) + [\mathbf{B}]^{\mathrm{T}}\frac{d[\mathbf{S}]}{d\mathcal{X}}[\mathbf{B}]\det([\mathbf{J}]) + [\mathbf{B}]^{\mathrm{T}}[\mathbf{S}]\frac{d[\mathbf{B}]}{d\mathcal{X}}\det([\mathbf{J}]) + [\mathbf{B}]^{\mathrm{T}}[\mathbf{S}][\mathbf{B}]\frac{d\,\det([\mathbf{J}])}{d\mathcal{X}} \Bigg) W_{GP}, \tag{41}$$

$$\frac{d[\mathbf{H}_e]}{d\mathcal{X}} = \sum_{GP} \Bigg(\frac{d[\mathbf{P}]^{\mathrm{T}}}{d\mathcal{X}}[\mathbf{C}]^{-1}[\mathbf{P}]\det([\mathbf{J}]) + [\mathbf{P}]^{\mathrm{T}}[\mathbf{C}]^{-1}\frac{d[\mathbf{P}]}{d\mathcal{X}}\det([\mathbf{J}] + [\mathbf{P}]^{\mathrm{T}}[\mathbf{C}]^{-1}[\mathbf{P}]\frac{d\,\det([\mathbf{J}])}{d\mathcal{X}} \Bigg) W_{GP}. \tag{42}$$

These equations require the unknown terms $\frac{d\mathbf{B}}{d\mathcal{X}}$, $\frac{d\mathcal{F}}{d\mathcal{X}}$, $\frac{d\,\det(\mathbf{J})}{d\mathcal{X}}$, $\frac{d\mathbf{H}_e^{-1}}{d\mathcal{X}}$, $\frac{d\mathbf{P}}{d\mathcal{X}}$, $\frac{d\mathbf{E}}{d\mathcal{X}}$, and $\frac{d\mathbf{S}}{d\mathcal{X}}$. The term $\frac{d\mathbf{H}_e^{-1}}{d\mathcal{X}}$ is calculated using the known relationship

$$\frac{d[\mathbf{H}_e]^{-1}}{d\mathcal{X}} = -[\mathbf{H}_e]^{-1}\frac{d[\mathbf{H}_e]}{d\mathcal{X}}[\mathbf{H}_e]^{-1}. \tag{43}$$

The terms $\frac{d\mathcal{F}}{d\mathcal{X}}$, $\frac{d\mathbf{S}}{d\mathcal{X}}$, and $\frac{d\mathbf{E}}{d\mathcal{X}}$ are calculated from the expressions

$$\frac{d[\mathcal{F}]}{d\mathcal{X}} = \frac{d[\mathbf{B}]}{d\mathcal{X}}\mathbf{u}_e, \tag{44}$$

$$\frac{d[\mathbf{S}]}{d\mathcal{X}} = \frac{d[\mathbf{P}]}{d\mathcal{X}}[\beta_e] + [\mathbf{P}]\frac{d[\beta_e]}{d\mathcal{X}}, \tag{45}$$

$$\frac{d\mathbf{E}}{d\mathcal{X}} = \frac{d[\mathcal{F}]}{d\mathcal{X}}\mathcal{F} + [\mathcal{F}]\frac{d\mathcal{F}}{d\mathcal{X}}. \tag{46}$$

The Jacobian term for 2D analysis is expressed as

$$[\mathbf{J}] = \begin{bmatrix} \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \\ \dfrac{\partial x}{\partial \zeta} & \dfrac{\partial y}{\partial \zeta} \end{bmatrix}, \tag{47}$$

where variables x and y are global coordinates and variables η and ζ are local element coordinates. The relationships between the these two sets of coordinates are expressed in the shape functions assumed for FEM. In the case where the isoparametric formulation is used, these shape functions are used for both the displacement and geometry. Equation (47) can then be rewritten as

$$[\mathbf{J}] = \begin{bmatrix} \dfrac{\partial N_1}{\partial \eta} & \dfrac{\partial N_2}{\partial \eta} & \dfrac{\partial N_3}{\partial \eta} & \dfrac{\partial N_4}{\partial \eta} \\ \dfrac{\partial N_1}{\partial \zeta} & \dfrac{\partial N_2}{\partial \zeta} & \dfrac{\partial N_3}{\partial \zeta} & \dfrac{\partial N_4}{\partial \zeta} \end{bmatrix} \mathcal{X}_e, \tag{48}$$

where the shape functions, N_1, N_2, N_3, and N_4 for a 4 noded element are expressed as

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\eta)(1-\zeta), \\ N_2 &= \frac{1}{4}(1+\eta)(1-\zeta), \\ N_3 &= \frac{1}{4}(1+\eta)(1+\zeta), \\ N_4 &= \frac{1}{4}(1-\eta)(1+\zeta). \end{aligned} \tag{49}$$

The determinant of $[\mathbf{J}]$ is described as

$$\begin{vmatrix} \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \\ \dfrac{\partial x}{\partial \zeta} & \dfrac{\partial y}{\partial \zeta} \end{vmatrix} = \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \zeta}, \tag{50}$$

where the terms in Equation (50) can easily be found from Equation (48). Therefore, the $\frac{d\,\det([J])}{d\mathcal{X}}$ term is described as

$$\begin{aligned} \frac{d\,\det([\mathbf{J}])}{dX_n^1} &= \frac{\partial N_n}{\partial \eta}\frac{\partial y}{\partial \zeta} + \frac{\partial N_n}{\partial \zeta}\frac{\partial y}{\partial \eta} \\ \frac{d\,\det([\mathbf{J}])}{dX_n^2} &= \frac{\partial N_n}{\partial \zeta}\frac{\partial x}{\partial \eta} + \frac{\partial N_n}{\partial \eta}\frac{\partial x}{\partial \zeta}. \end{aligned} \tag{51}$$

The terms $\mathcal{X}_n^d$ denote the element node number n and d indicates the degree-of-freedom.

The strain-displacement term, $[\mathbf{B}]$, is described as

$$[\mathbf{B}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} [\mathbf{J}]^{-1} & [0] \\ [0] & [\mathbf{J}]^{-1} \end{bmatrix} [\mathbf{T}], \tag{52}$$

where $[\mathbf{T}]$ is only a function of the local co-ordinate variables. Therefore, the derivative of this equation is then simply

$$\frac{d[\mathbf{B}]}{d\mathbf{D}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} [\frac{d[\mathbf{J}]^{-1}}{d\mathcal{X}} & [0] \\ [0] & \frac{d[\mathbf{J}]^{-1}}{d\mathcal{X}} \end{bmatrix} [\mathbf{T}], \tag{53}$$

as both the constant matrix and the term $[\mathbf{T}]$ are independent of the nodal coordinates of the element. The final remaining unknown term, $\frac{d[\mathbf{J}]^{-1}}{d\mathcal{X}}$, is then calculated using

$$\frac{d[\mathbf{J}]^{-1}}{d\mathcal{X}} = -[\mathbf{J}]^{-1} \frac{d[\mathbf{J}]}{d\mathcal{X}} [\mathbf{J}]^{-1}, \tag{54}$$

where the term $\dfrac{d[\mathbf{J}]}{d\mathcal{X}}$ is expressed as

$$\begin{aligned} \frac{d[\mathbf{J}]}{d\mathcal{X}_n^1} &= \begin{bmatrix} \dfrac{\partial N_n}{\partial \eta} & 0 \\ \dfrac{\partial N_n}{\partial \zeta} & 0 \end{bmatrix} \\ \frac{d[\mathbf{J}]}{d\mathcal{X}_n^2} &= \begin{bmatrix} 0 & \dfrac{\partial N_n}{\partial \eta} \\ 0 & \dfrac{\partial N_n}{\partial \zeta} \end{bmatrix}. \end{aligned} \tag{55}$$

In summary, the total sensitivity procedure can be separated into to two basic sub-procedures. The first sub-procedure, Sub-procedure 1, calculates the internal residual derivative and is dependant on both the implemented element and meshing scheme.

The other sub-procedure, Sub-procedure 2, incorporates the first sub-procedure and is dependant on the ALC method. The implementation of these sub-procedures creates an analytical sensitivity procedure for highly geometrically non-linear structures.

Sub-procedure 1: Residual Derivative for Assumed Stress Element

Result: $\frac{d\mathbf{R}}{d\mathbf{x}}$
Calculate $\frac{d\mathcal{X}}{d\mathbf{x}}$;
for *All Degrees of Freedom in the Mesh* **do**
  Compute Equation (55);
  Use this result and Equation (54) to compute Equation (53);
  Compute Equation (51);
  Determine Equations (36) and (37) using Equations (38) to (42);
  for *All Design Variables* **do**
    Multiply the result by the appropriate component in the $\frac{d\mathcal{X}}{dx}$ matrix;
    Assemble this component into the $\frac{d\mathbf{R}}{dx}$ matrix;
  end
end

Sub-procedure 2: ALC Analytical Sensitivity

Result: $\frac{d\mathbf{u}}{dx}$ and $\frac{d\lambda}{dx}$ for each solution step.
Set $\frac{d\mathbf{u}}{dx}$ and $\frac{d\lambda}{dx}$ to 0;
for *End of Each Solution Steps* **do**
  Complete Algorithm 1;
  Solve Equation (27);
  Accumulate the $\frac{d\mathbf{u}}{dx}$ and $\frac{d\lambda}{dx}$ vectors;
end

B Ethics Declarations

B.1 Conflict of Interest

The authors declare that they have no conflict of interest.

B.2 Replication of Results

All necessary algorithms and problem parameters for possible replication of all results presented in this work have been detailed and referenced.