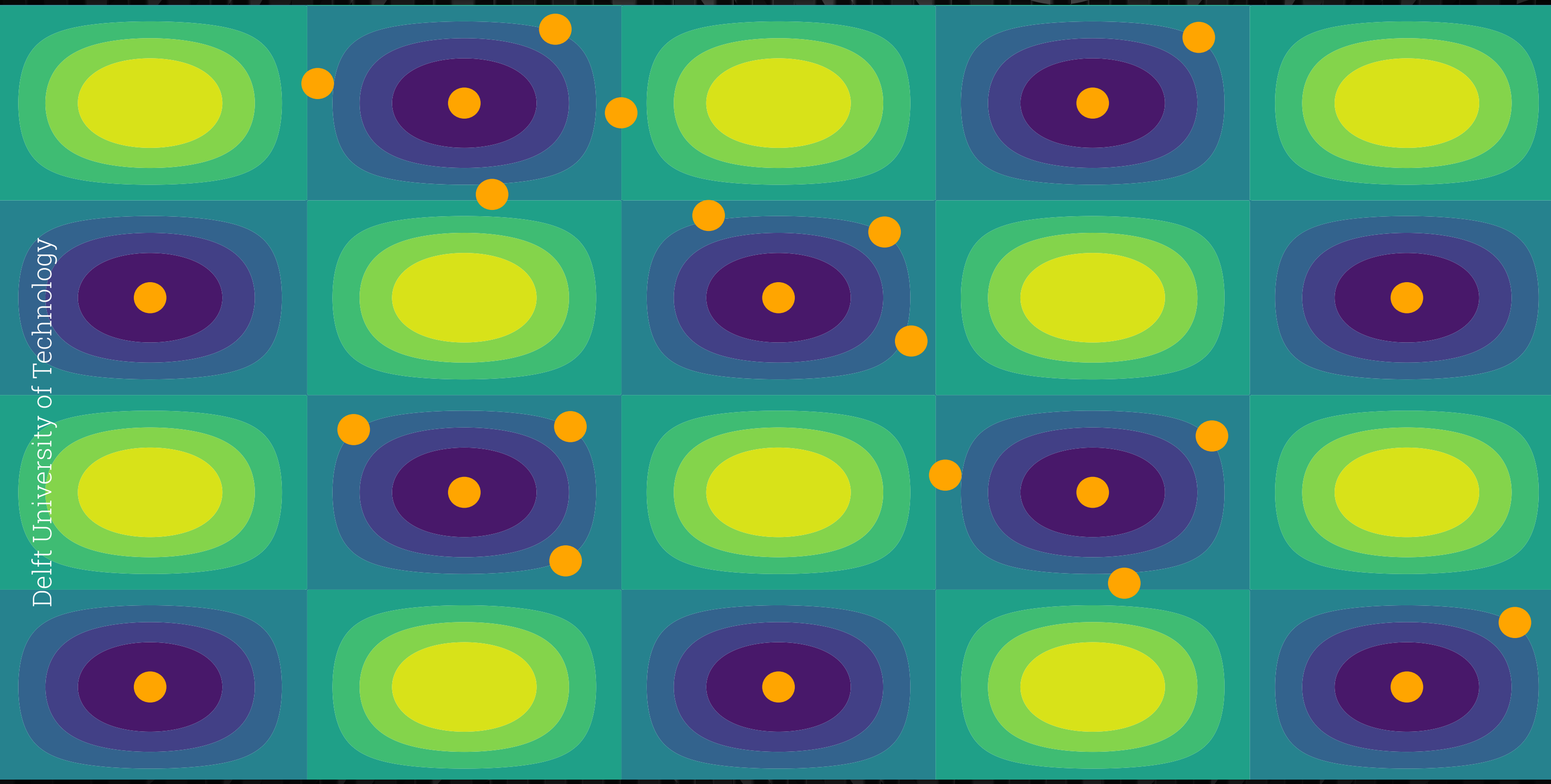
Global optimization using a deflation-based method for the design of composite structures

MSc. Thesis

Sankalp Seetharama Bangera



Delft University of Technology

TU Delft

Global optimization using a deflation-based method for the design of composite structures

by

Sankalp Seetharama Bangera

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday, March 27, 2023 at 9:00 AM.

Student number:	5336244	
Project duration:	March 1, 2022 – March 27, 2023	
Thesis committee:	Dr. S.G.P. Castro,	TU Delft, Supervisor and chair
	Dr. D.M.J. Peeters,	TU Delft, examiner
	Dr. J.A. Pascoe,	TU Delft, examiner
	Dr. O.K. Bergsma,	TU Delft, examiner

This page was intentionally left blank.

Preface

"Climbing to the top demands strength, whether it is to the top of Mount Everest or to the top of your career."

A. P. J. Abdul Kalam.

This thesis is written to obtain my Master's degree in Aerospace Engineering with a specialization in aerospace structures at Delft University of Technology, Netherlands.

The fall of 2020, was when I came to the Netherlands. The timing back then seemed to be alright since the Covid-19 pandemic numbers were set to drop towards the end of the year with the resumption of normal lectures, however, this was not the case as I finished my compulsory course quarters by mid-2021. Even though most of the education I obtained was not in a lecture hall, the knowledge and experience I have gained are personal achievements I am proud of, and I am glad I decided to travel for my studies.

The phase of working on my thesis has had multiple lows and highs, much like the functions I have been optimizing. And through all these lows and highs I have had the support and encouragement of some people who have helped me grow stronger both, mentally and physically. These people I will forever cherish.

I would like to extend my immense gratitude, to my thesis supervisor Prof. Saullo Castro from Tu Delft, who has provided me invaluable academic guidance throughout my thesis project with all the meetings and discussions we have had, and for being supportive throughout this journey. The role he has played during my thesis project is actually analogous to the role of the optimizer I have proposed in the project; nudging me step by step to follow a path leading to the optimal point.

My mother (Krapa Bangera), father (Seetharama Bangera), and brother (Sanket Bangera) have been my pillars of support throughout my life, and the work I have put in through all these years be it academics or life, in general, is dedicated to their belief in me. This journey would have been a lot harder in the absence of their love and support and I am extremely grateful to them for providing me with this life full of opportunities.

I have three of my close friends I would like to express my gratitude, for making me feel the comfort of home despite being so far from it. Shikha Mahboobani is the person who has virtually graced most of my days away from home by being my personal cheerleader and day-to-day logbook. Ahaan Bhosale, a fellow batch-mate since 2016, has been like a brother to me, and being enrolled in the same course at the same time helped both of us to cope with mental fatigue. He has been a motivator for my physical strength too, thanks to all the nighttime gym sessions. Radha Dave is the person who provides me with a good laugh from time to time and also shares a similar path to mine. They have supported me invaluably through this journey and I am thankful to them for always being there for me.

All these people have contributed to my thesis in more ways than I can not express, and for that, I am grateful to all of them.

Sankalp Seetharama Bangera
Delft, March 2023

This page was intentionally left blank.

Abstract

Composite structures are rapidly transforming the aerospace industry, driven by continuous advancements in manufacturing methods capable of producing optimized structures with variable stiffness that enables the creation of increasingly complex and efficient structures. This project focuses on the development of a global optimization method that applies the innovative concept of deflation to the design of optimized composite structures.

This project aims at developing a method for global optimization by applying the concept of deflation for the design of optimized composite structures. Gradient-based optimization is known for its accuracy in identifying local optima, although heavily depends on initial starting points in non-convex design spaces. By incorporating deflation, gradient-based optimizers can obtain multiple local optima even when starting from the same point in the design space. This approach not only offers alternate minima for assessing design feasibility but also highlights the importance of having a universally applicable method for existing optimization schemes.

The methodology herein proposed establishes a gradient-based optimization framework that is used to develop and test the developed deflation constraint. The novel deflation constraint can be integrated into any optimization method supporting constrained optimization, either gradient-based or heuristics-based. The developed deflation method is tested on various case studies related to composite structure optimization, showcasing its promising applications in the aerospace industry and beyond.

This page was intentionally left blank.

Contents

This page was intentionally left blank.

Acronyms

AD	Automatic Differentiation
AFP	Automated Fiber Placement
ATL	Automated Tape Laying
CA	Cellular Automation
CTS	Continuous Tow Shearing
KKT	Karush Kuhn Tucker
LP	Lamination Parameters
NLP	Non-Linear Programming
PDE	Partial Differential Equation
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
VS	Variable Stiffness

This page was intentionally left blank.

1 Introduction

1.1. Context and motivation

Structural optimization is a fairly vast subject in itself and is usually associated with lightweight structures. Composites have been a major advent in exploring these lightweight structures owing to their superior specific material properties (ratio with respect to weight) as compared to metals. In the aircraft industry, Boeing was the first to enter an aircraft with a composite fuselage and composite wings with about 50% structure contributing to significant improvement in efficiency (increased range, decreased operation costs, etc.) [1]. Although a significant improvement over metallic aircrafts has been achieved, there still exists quite a bit of untapped potential in composites.

Composites structures/laminates are composed of layered fiber-reinforced material with each layer having its fibers oriented in a particular direction. These layers could be composed of the same materials or even different. The composites considered in this work have a consistent material. The mechanical properties of composite laminates depend on the orientation of their layers. Hence the fiber orientations can be tailored to optimize the laminate for particular service conditions. This tailor-ability is a powerful suite of composite structures.

Laminates associated with layers that have straight and parallel fibers are known as constant stiffness laminates as the stiffness properties across the laminate are constant. Having these fibers straight and parallel but optimized for a particular application still hides quite a bit of its potential. This is where Variable Stiffness (VS) laminates come in. VS is achieved by the spatial variation of the stiffness of laminate [2]. Hence, if the varying spatial stress state in a laminate for a particular application is known, the fiber orientation angles for each layer can be varied which results in more efficient usage of directional properties. VS is no more a theoretical concept and is achievable with current composite manufacturing technologies. However, the study of optimization of VS structures could still benefit from the advent of new optimization methods to optimize these structures from a global perspective because the design space associated with the spatial orientation of fiber angles can be highly non-convex [3].

For global optimization, a method that can explore multiple local minima to find a global minimum is desirable when dealing with a non-convex design space. Many of these optimizers rely on different sampling or starting point approaches [4], [5], however, the methods do not guarantee not converging onto a solution that has been found previously. This issue is remedied by a method known as deflation, which guarantees the discovery of new distinct minima even while starting from the same starting point. Hence this leads to the topic of the present thesis;

Global optimization using a deflation-based method for the design of composite structures.

Gradient-based optimization is known for its ability to recover local minima with high accuracy [6]. This method coupled with deflation would help gradient-based optimizers become global optimizers as well. The aim of this project involves the development of a methodology for a gradient-based optimization algorithm, followed by the formulation of a method for deflation that can be applied to any optimization algorithm that supports constrained optimization. The development of such a method for deflation will be one of the major contributions of this work. The deflation method on its when applied to an optimization algorithm involves significant changes that need to be made to the algorithm, but if it is formulated as a constraint, deflation can be applied without altering the optimization algorithm and by just adding a constraint to the optimization problem.

1.2. Research questions

Based on the requirements that this thesis needs to accomplish, the primary research question is framed as,

Primary Research Question

How can a deflated gradient-based optimization method be created that can be used for global optimization of the non-convex design spaces associated with composite structural design problems?

This primary research question can be further divided into multiple research sub-questions. Firstly, emphasis is laid on the structure of the optimization algorithm and the following questions are framed:

Research Sub-Questions

How can a gradient-based optimization algorithm be created for constrained optimization?

1. How are constraints included in an optimization algorithm?
2. Which gradient-based optimization method is suitable for the selection of search (descent) direction?
3. What is a line-search method and how can a line-search algorithm be created?
4. How can the gradients involved in gradient based-optimization be evaluated efficiently without utilizing symbolic differentiation?

Secondly, emphasis is laid on the deflation method and the following questions are framed:

Research Sub-Questions

How can deflation be applied to optimization algorithms to obtain multiple minima?

1. What are the aspects and major parameters of the deflation method?
2. Can a method be devised to enable easy implementation of deflation to any constrained optimization algorithm?

Lastly, emphasis is laid on how the optimization methodology can be implemented for the design of composite structures, and the following questions are framed:

Research Sub-Questions

Can the optimization of composite structures be achieved with the proposed method?

1. Can the stacking sequence of constant stiffness laminates be optimized with the proposed method?
2. What are the different ways of parameterizing VS Laminates?
3. How can spatial fiber orientation angles be retrieved from optimized Lamination parameters?

1.3. Report layout

The report is divided into three parts, with each part having certain chapters. The first part of the report consists of the literature review. This is further divided into three chapters; chapter 2 covers the basic literature required to understand gradient-based optimization and the proposed methodology, chapter 3 covers the basic concept of the deflation method, and chapter 4 covers the concept of VS laminates, along with aspects like manufacturing, defects and design parameterization. The second part of the report consists of the methodology and is further divided into two chapters; chapter 5 covers the development of the optimization algorithm for this project and chapter 6 covers the proposed method of using a deflation constraint that can be implemented in any constrained optimization algorithm. The last part of this report includes the results obtained with the proposed methodology; chapter 7 covers the various case studies that the methodology is tested on, followed by chapter 8 where the research questions are revisited and provide summaries to each question based on which future recommendations are proposed and lastly chapter 9 which summarizes the key contributions this work.

This page was intentionally left blank.

Part I

Literature Review

2 Optimization

This chapter covers the basic concepts of optimization that are the basis of this thesis. The chapter discusses the main types of optimization problems, gradient-based optimization, line search, the inclusion of constraints based on Lagrange multipliers, penalty methods, and automatic differentiation (also known as algorithmic differentiation). In this thesis the Optimization algorithm is designed to handle continuous objectives and the literature covered in this chapter supports the same. Martins and Ning provide extensive Literature for Optimization in their book [6] and is one of the primary references for this chapter.

2.1. Representations

The mathematical representation of an optimization problem is as follows [6],

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{by varying} & \underline{x}_i \leq x_i \leq \overline{x}_i & i = 1, \ldots, n_x \\ \text{subject to} & g_j(x) \leq 0 & j = 1, \ldots, n_g \\ & h_i(x) = 0 & l = 1, \ldots, n_h. \end{array} \tag{2.1}$$

where $f(x)$ is the scalar objective function in terms of the design variable x, n_x is the number of design variables, $\underline{x}_i$ is the lower limit of x, $\overline{x}_i$ is the upper limit of x, $g_j(x)$ is the inequality constraint in an implicit form with n_g being the number of inequality constraints, and lastly $h_l(x)$ is the implicit equality constraint with n_h being the number of equality constraints. This mathematical representation is a common format encountered in literature and is herein adopted, being interpreted (Equation 2.1) as follows: "*minimize the objective function by varying the design variables within their bounds subject to the constraints*" [6].

The mathematical models used in certain optimization algorithms can scale up in complexity and representing them with explicit relations becomes difficult. However these mathematical models can be expressed in an implicit state i.e. the residual formation given as [6],

$$r_i(u_1, ..., u_n) = 0, i = 1, .., n, \tag{2.2}$$

where r is a vector of residual with the size being the same as the state variables u. The equations in any mathematical model can be represented as a set of equations in this form which is more compactly written as $r(u) = 0$. Figure 2.1 depicts how the optimization algorithm functions. First, the optimizer

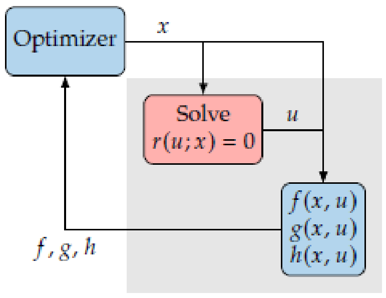
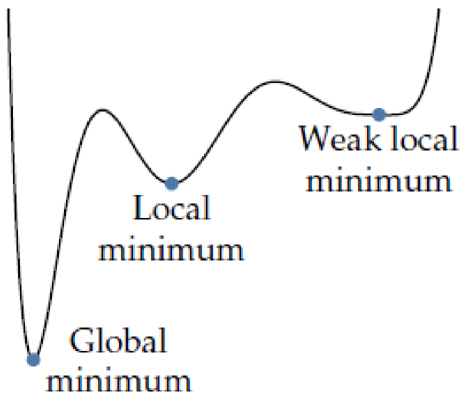


Figure 2.1: Computation of objective function and constraints with new state variables obtained from residuals [6]

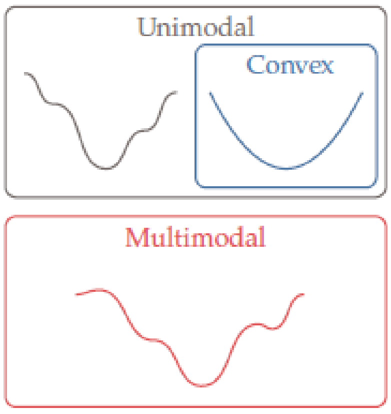
sends the initial design variables x to the solver to generate new state variables after which the design variables and state variables are used to obtain the values of the objective function and constraints. Earlier in Equation 2.1 the governing equations were assumed to be part of the computation, and hence the objective function and constraints were written only in terms of the design variables. With these evaluated values, sent to the optimizer, a new point in the design space is found and the cycle continues until the optimizer converges on a minimum.

2.2. Convex and non-convex functions

The modality of a function is a quality that helps in determining if the function is convex or non-convex. Unimodal functions have a single minimum while multimodal functions contain multiple minima [6]. A minimum can be labeled as a global minimum only if the function is known to be unimodal. To prove that the function is unimodal would require evaluating the function through all points in the domain and this can be computationally expensive. A way to prove unimodality is by optimizing the problem multiple times from different starting points. With increasing iterations, if the optimization converges to the same minimum the confidence of the function being unimodal increases. For instance, He et al. [7] and Lyu et al. [8] in their respective works on an aerodynamic shape optimization problem shows how the optimizer converges to the same optimum despite choosing different starting points. If two or more distinct minima are found from different starting points, then the function is multimodal. In Figure 2.2a, three minima are distinguished. When observed in narrow ranges, each of these minima can be evaluated as local minimums but when viewed over the entire range, the minimum that lies lower than the others is the global minimum, while the others are local minima. There is also a third type there, which is a weak local minimum and is called so because it is a minimum that is located above the rest of the minima.



(a) Types of minima.



(b) Unimodal and multimodal functions.

Figure 2.2: Modality of functions [6].

Convexity requires that in a function, all line segments connecting any two points in the function are above the function and do not intersect it. Hence not all unimodal functions are convex as can be seen in Figure 2.2b, but all multimodal functions are non-convex. When convex functions are dealt with, the optimization of such functions is straightforward and methods like gradient descent [9], [10] can be used to evaluate the global minimum with high accuracy. As the design space becomes increasingly multimodal or non-convex, evaluating a global minimum becomes more difficult and would require special techniques to evaluate multiple points in the design space to find the point that best minimizes the function. However, this may not be the global minimum. This thesis project aims at creating an optimization method that can facilitate locating a global minimum through deflation, and further chapters explain the same.

2.3. Gradient-based optimization

In this section, the basics of gradient-based optimization are covered and should provide sufficient information to interpret the content of chapter 5.

As the name implies, gradient-based methods rely on the gradient information of the objective function and the respective constraints for minimization, by effectively moving in the direction of function decrease.

2.3.1. Fundamentals

To understand the basics of this method it is important to have the fundamentals in order. It is seen previously in Equation 2.1, a scalar objective function $f(x)$ is considered, in which $x = [x_1, ..., x_n]$ is the vector of design variables. The gradient of the objective function is given as [6],

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n}\right] \tag{2.3}$$

where the gradient of the objective function contains components that quantify the function's rate of change with respect to a particular design variable which is the slope of the function given in the direction of each design variable. The gradient depicts a vector that points in the direction of the steepest function increase from a particular point.

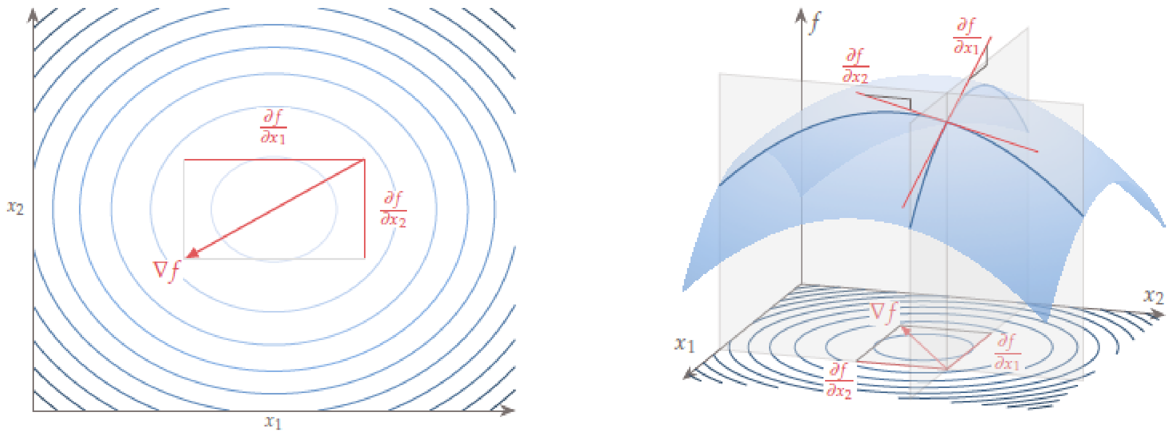

$\frac{\partial f}{\partial x_1}$
$\frac{\partial f}{\partial x_2}$
∇f
x_1
x_2
f

Figure 2.3: Components of gradient vector in a two-dimensional case [6]

Figure 2.3 depict the gradient vector ∇f along with the components corresponding to x_1 and x_2 directions. In this 2D case, the gradient vectors taken at any point are always perpendicular to the function's contour lines. In an n-dimensional space (isosurfaces), the gradient vectors are normal to the surfaces of constant f [6].

Usually, in optimization algorithms, it is required to have the gradient information described in a particular direction and is given by a directional derivative. Hence this derivative can be obtained by projecting the gradient vector onto the desired direction p as follows,

$$\nabla_p f(x) = \nabla f^\top p = \|\nabla f\| \|p\| cos\theta \tag{2.4}$$

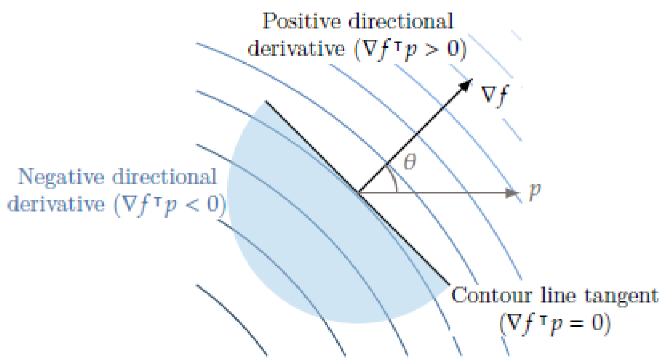

Positive directional derivative ($\nabla f^\top p > 0$)
∇f
θ
Negative directional derivative ($\nabla f^\top p < 0$)
p
Contour line tangent ($\nabla f^\top p = 0$)

Figure 2.4: Directional derivative in the direction p is given $\nabla f^\top p$ [6]

The Figure 2.4 shows the direction vector p along which the directional derivative $\nabla_p f(x)$ lies. The angle between the gradient vector ∇f and the direction vector p is denoted as θ. When θ is in the interval $(-90, 90)°$, the directional derivative is positive and is therefore pointed in the direction of function increase. If θ is between $(90, 270)°$ the directional derivative is negative and is hence directed in the direction of function decrease. Lastly, if $\theta = \pm 90$, the function value does not change significantly for small steps in this direction as it is locally flat. Hence, when ∇f and p are orthogonal this condition takes place. and it can be inferred that the gradient is orthogonal to the function isosurfaces [6].

The curvature of a function provides useful information by describing the rate of change of the gradient of the function. So, positive curvature means the slope of the function is increasing, negative curvature means the slope is decreasing, and zero curvature which is stationary. Considering a function with n-variables, the derivatives of all gradient components with respect to all gradient directions yield a second-order square-tensor known as the Hessian and is given as [6],

$$H_f(x) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}, \tag{2.5}$$

The Hessian can also be expressed with index notation as,

$$H_{f_{ij}} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{2.6}$$

In the Hessian, each row i is a vector that gives the rate of change of all the j components with respect to the ith direction. While each column j corresponds to the rate of change of component j with respect to all coordinate directions i. Due to the property of symmetry of second derivatives, the Hessian is a symmetric matrix with $n(n+1)/2$ independent entries. Now, when a direction p is considered, the rate of change of gradient along that direction is obtained by the product Hp, The curvature of the one-dimensional function along the arbitrary direction p is then given as [6],

$$\nabla_p(\nabla_p f(x)) = p^\top Hp \tag{2.7}$$

An n-dimensional Hessian can also be formulated into an eigenvalue problem given as [6],

$$H\nu = \kappa\nu \tag{2.8}$$

where the eigenvectors ν represents the principal curvature directions, and the eigenvalues κ quantify the curvatures. To further visualize and understand gradients and curvatures, the reader can find a two-variable example in Appendix A.

2.3.2. Optimality conditions

For a certain point x^* to be the minimum of a function (in this case a local minimum, because global optimization of a non-convex function is not possible with just gradient-based methods), it should be such that $f(x^*) \leq f(x)$ for all the values of x that lie around x^*. A second-order Taylor series expansion can be used to calculate the function value across any direction p and is given as [6],

$$f(x^* + p) = f(x^*) + \nabla f(x^*)^\intercal p + \frac{1}{2}p^\intercal H(x^*)p + \dots . \tag{2.9}$$

For x^* to be a minimum, it must satisfy that $f(x^* + p) \geq f(x^*)$ for all p. Thus, it can be implied that the first and the second order terms in Equation 2.9 should be greater than 0 and is,

$$\nabla f(x^*)^\intercal p + \frac{1}{2}p^\intercal H(x^*)p \geq 0 \quad \text{for all} \quad p. \tag{2.10}$$

Observing the first term in this equation (Equation 2.10), p can be in any arbitrary direction and the only way to satisfy the inequality is that all elements of the gradient vector need to be zero (refer Figure 2.4) and is given as,

$$\nabla f(x^*) = 0 \tag{2.11}$$

This is the first-order necessary optimality condition in an unconstrained problem. In a constrained problem, this condition would have to be satisfied by the Lagrangian of the function, but more on that is covered in the later sections. Now because the gradient term in Equation 2.10 needs to be zero, the other term involving the Hessian can be considered in the inequality. This term represents the curvature term as seen previously in Equation 2.7 which means that the curvature of the function must be positive or zero in any projected direction p. Hence the inequality now resembles the requirement of a positive-semidefinite matrix and is given as [6],

$$p^\intercal H(x^*)p \geq 0 \quad \text{for all} \quad p. \tag{2.12}$$

This requires the Hessian $H(x^*)$ to be positive semidefinite, which implies that the eigenvalues of the Hessian must be greater than or equal to zero. However the conditions for gradient and curvature may be insufficient if the curvature is zero in some direction p, and the only way to know if x^* is a minimum

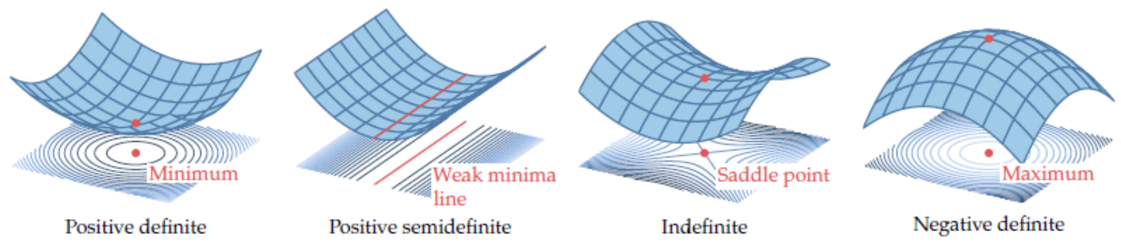


Figure 2.5: Quadratic functions with different types of Hessians [6].

is to check the third order term in the Taylor series (Equation 2.9) hence resulting in a weak minimum. The different optimum points resulting from Hessians of different quadratic functions are depicted in Figure 2.5.

2.3.3. Line Search

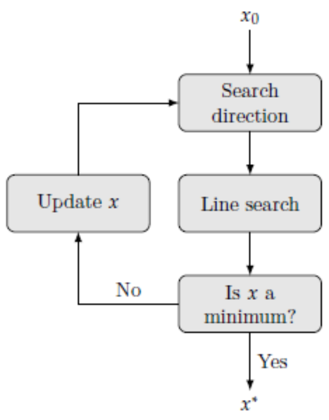


Figure 2.6: Optimization using a line search process flow [6].

When the gradient information regarding function is known, and the direction of decrease is identified (using methods like steepest descent, conjugate gradient, Newton's method, etc. [6]), a line search determines the sufficient step that needs to be taken in that direction to get closer to the minimum, without overshooting it. The Figure 2.6 shows the three main steps in an algorithm involving line search. After an appropriate step is selected, the variables are updated and if the minimum has not been reached, the process is repeated [6].

As seen in Figure 2.6, the two main sub-problems are, choosing the direction of search, and determining how far to step in that direction. The former usually determines the name of the optimization process, and the one used for this thesis is covered in chapter 5, but here the step that is covered is selection of how far to step in a particular search direction. After stepping a certain distance in direction descent direction p_k (which guarantees decrease in function value f) for iteration k, the design variable is updated as,

$$x_{k+1} = x_k + \alpha p_k \tag{2.13}$$

where α is always positive and represents the step in the direction p_k. It should be noted that in this case, the goal is to not minimize $f(x_k + \alpha p_k)$ but to step enough that results in a good decrease in function value. This is because finding the exact minimum would require many computations and is considered wasteful [6]. Some new notation is introduced to simplify the expressions that lie ahead.

$$\begin{aligned}\phi(\alpha) &= f(x_k + \alpha p_k)\\ \phi'(\alpha) &= \nabla f(x_k + \alpha p_k)^\top p_k\end{aligned} \tag{2.14}$$

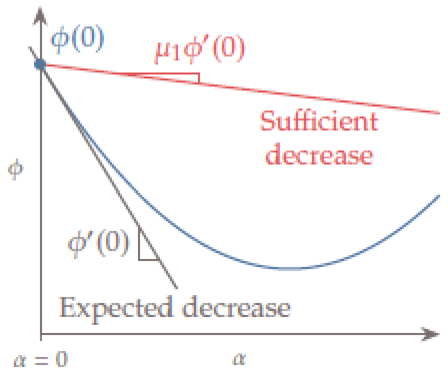
$phi'(\alpha)$ represents the directional derivative along the search direction. $\phi'(0)$ represents the slope at the beginning of the line search and must always be negative since p_k is a descent direction. Hence the line search method used in an optimization method contributes to the efficiency of the method. Martins and Ning [6] cover two line search algorithms which are based on the following conditions,

Sufficient decrease condition:

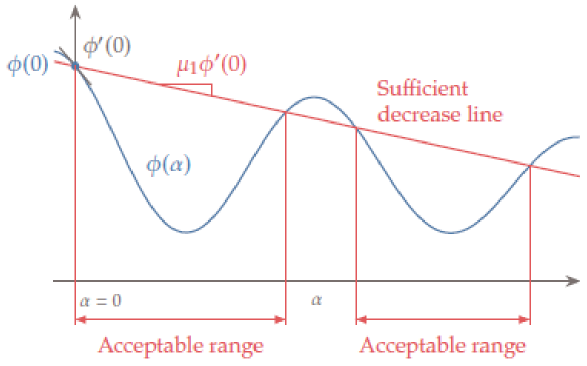
The sufficient decrease condition (also known as Armijo condition), is mathematically given as [6],

$$\phi(\alpha) \leq \phi(0) + \mu_1 \alpha \phi'(0), \tag{2.15}$$

where μ_1 is a constant that varies between $(0, 1)$. In Equation 2.15, $\alpha\phi'(0)$ indicates the expected decrease in function value by assuming that the function continues along the same slope. While the constant μ_1 is multiplied to $\alpha\phi'(0)$ and indicates that Equation 2.15 is satisfied even if a fraction of the expected decrease is achieved as shown in Figure 2.7a.



(a) The slope of sufficient decrease line is a fraction of the slope $\phi'(0)$.



(b) Acceptable ranges for α.

Figure 2.7: The sufficient decrease condition [6].

In a line search, the slope at the starting point ($\alpha = 0$) is known, but how the function varies is unknown until it is evaluated. Hence the first point that is encountered in a line search that is below the sufficient decrease line is selected as α as shown in Figure 2.7b. Based on this condition an algorithm known as backtracking (Algorithm 1) is proposed. The algorithm starts with a maximum step ($\alpha_{init} = 1$, if slack variables are not involved) in the direction of descent and is reduced in successive iterations by a constant ratio ρ (typically 0.5) until the sufficient decrease condition is satisfied. The algorithm is as follows [6],

The backtracking line search is quite a straightforward method but there are a few scenarios where this performs poorly which are stated as follows,

1. Firstly, in a certain scenario, the initial guess is far too large, and in attempts to satisfy the sufficient decrease, the step length becomes smaller than the initial one by several orders of magnitude. Hence if ρ is not chosen carefully, it can result in many backtracking evaluations.
2. Secondly, if the initial guess were to immediately satisfy the sufficient decrease condition, that value is chosen. Whereas if it had to step further, it could reduce the function value more.

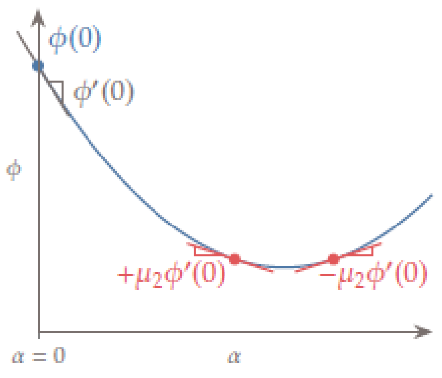
Algorithm 1 Backtracking line search

```
Inputs:
  $\alpha_{init} > 0$                                  ▷ Initial step length
  $0 < \mu_1 < 1$                    ▷ Sufficient decrease factor, typically $10^{-4}$
  $0 < \rho < 1$                          ▷ Backtrack factor, (eg. $\rho = 0.5$)

Output:
  $\alpha^*$                 ▷ Step size satisfying sufficient decrease condition

$\alpha = \alpha_{init}$
while $\phi(\alpha) > \phi(0) + \mu_1 \alpha \phi'(0)$ do   ▷ Function value is above sufficient decrease line
    $\alpha = \rho\alpha$                                               ▷ Backtrack
    Evaluate $\phi(\alpha)$
end while
```

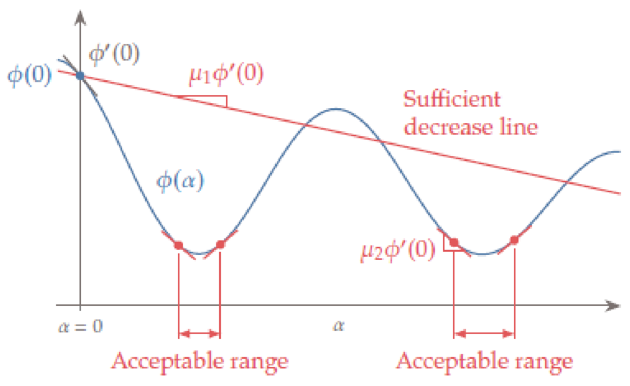
3. The constant reduction factor ρ provides no flexibility and could be adjusted to be a dynamic factor based on the function values and gradients to make more intelligent evaluations.

Despite these shortcomings, the simple method of backtracking provides a quick line search and for this thesis proved to be the method of choice. The reason is that it is a robust method that makes the optimizer run quicker as compared to another algorithm that was tested. The difference in speed grew as the size of the design space increased.

Strong Wolfe Conditions



(a) The sufficient curvature condition implies the slope of the function to be a fraction of the initial $\phi'(0)$.



(b) Acceptable ranges for α.

Figure 2.8: The strong Wolfe conditions [6].

The Strong Wolfe conditions are meant to fill the gaps left in the sufficient decrease condition. A weakness of the sufficient decrease condition alone is that it accepts a small step that satisfies the condition for a marginal decrease in function value. This can be improved by increasing μ_1 in Equation 2.15 to result in a greater decrease with a larger step size, but it still avoids a larger step size that would result in a further reasonable decrease. Therefore in addition to the sufficient decrease condition, another condition can be added, with which, how much of a function decrease can be expected is known. Hence comparing the slopes at the start of the line search and the candidate point, it can be figured if the function is flattening or not i.e. comparing the rate of change of the function as shown in Figure 2.8a. This is given by the sufficient curvature condition [6]:

$$|\phi'(\alpha)| \leq \mu_2 |\phi'(0)|. \tag{2.16}$$

μ_2 values usually range in $(0.1, 0.9)$ and the best value is quite problem dependent. As $\mu_2 \to 0$, the sufficient curvature condition would enforce an exact line search where $\phi'(\alpha) = 0$. The sufficient decrease and the sufficient curvature condition together form the strong Wolfe conditions. The acceptable range of α given by the strong Wolfe conditions is shown in Figure 2.8b and is more restrictive than in the case of the sufficient decrease condition (Figure 2.7b).

It must be noted that the sufficient curvature slope must be deeper than the sufficient decrease slope, i.e. $0 < \mu_1 < \mu_2 < 1$. If this is not followed, the line search could result in points where the strong Wolfe conditions are not satisfied and would look like the case shown in Figure 2.9. Martins and Ning propose an algorithm in their book to perform line search while satisfying the strong Wolfe conditions, however, it was not implemented in this project, and interested readers can refer to it in[6]. After testing both, the backtracking algorithm and algorithm satisfying the strong Wolfe conditions, it was found that the new algorithm did not provide significant improvement in the results obtained, however, the number of iterations required to find search directions by the optimizer was reduced. Although this decrease in iterations did not result in any computational time savings and was slower than the run utilizing the backtracking line search.

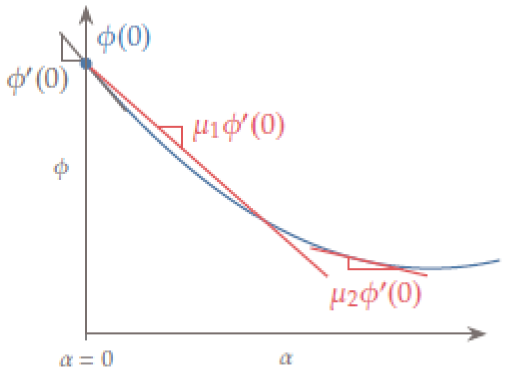


Figure 2.9: If $\mu_2 < \mu_1$ [6].

The slower computational time can be attributed to the fact that the Algorithm required the calculation of the slope at the new candidate points in addition to the function values, and also there were many conditional statements to be evaluated. Hence the simple structure of the backtracking algorithm gives it a significant advantage in computational speed.

2.4. Constrained gradient-based optimization

In the previous section, the basics involved in a gradient-based optimization algorithm were covered. In This section, the inclusion of constraints into gradient-based optimization is covered. Martins and Ning [6] provides a nice way to visualize the vector spaces of the involved gradient vectors of the objective along with the constraints and derive proofs for the modified optimality conditions (modified as compared to the ones seen in Equation 2.11 and Equation 2.12). The formulation of the optimization problem is shown in Equation 2.1. But before proceeding further it is important to know another terminology, i.e. the Jacobian or Jacobian matrix. The Jacobian is a matrix that gives the gradient information about certain residuals used in optimization. for instance, the first row of the Jacobian contains the gradients of the first residual with respect to all the design variables, the second row contains gradients for the second residual, and so on. It is given as,

$$J_{ij} = \frac{\partial r_i}{\partial x_j} \tag{2.17}$$

where r_i are the residuals and x_j the design variables. Considering an optimization problem as shown in Equation 2.1, the Lagrangian function can be used to convert a constrained problem to an unconstrained problem by introducing new variables known as Lagrange multipliers. The Lagrangian is given as [6],

$$\mathcal{L}(x, \lambda, \sigma, s) = f(x) + \lambda^{\mathsf{T}} h(x) + \sigma^{\mathsf{T}} \left(g(x) + s \odot s \right) \tag{2.18}$$

In this equation, a few new terms can be observed in addition to the objective function and constraints. λ is the vector of Lagrange multipliers that corresponds to the equality constraint vector, σ is the Lagrange multiplier vector that is associated with the inequality constraint vector and s is a vector of slack variables with $\odot$ symbol representing element-wise multiplication. The slack variables are associated with the inequality constraints, such that the inequality constraints are converted to equality constraints given by,

$$g_j + s_j^2 = 0, \quad j = 1, \ldots, n_g, \tag{2.19}$$

In an optimization problem, if both, equality and inequality constraints are present, the conditions associated with the inequality constraint apply only in the subspace of directions feasible with reference to the equality constraints and it is not known which inequality constraint may be active. Hence the significance of the slack variable is that when $s_j = 0$, the corresponding inequality constraint is active ($g_j = 0$), and when $s_j \neq 0$, the inequality constraint would be inactive. In Equation 2.19 the slack variable is squared to ensure non-negative values since Equation 2.19 can be satisfied only when the inequality constraint, $g_j \leq 0$ [6].

The optimality conditions can be derived by seeking a stationary (zero curvature) point in the design domain by differentiating the Lagrangian (Equation 2.18) with respect to all the variables and is done as follows. First, it is differentiated with respect to the design variables x [6],

$$\nabla_x \mathcal{L} = 0 \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_{l=1}^{n_h} \lambda_l \frac{\partial h_l}{\partial x_i} + \sum_{j=1}^{n_s} \sigma_j \frac{\partial g_j}{\partial x_i} = 0, \quad i = 1, ..., n_x \tag{2.20}$$

Secondly, taking the derivative of the Lagrangian with respect to the equality Lagrange multipliers [6],

$$\nabla_\lambda \mathcal{L} = 0 \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial \lambda_l} = h_l = 0, \quad l = 1, \ldots, n_h, \tag{2.21}$$

Thirdly, taking the derivative with respect to the inequality Lagrange Multipliers [6],

$$\nabla_\sigma \mathcal{L} = 0 \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial \sigma_j} = g_j + s_j^2 = 0 \quad j = 1, \ldots, n_g, \tag{2.22}$$

Lastly, taking the derivative of the Lagrangian with respect to the slack variables,

$$\nabla_s \mathcal{L} = 0 \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial s_j} = 2\sigma_j s_j = 0, \quad j = 1, \ldots, n_g, \tag{2.23}$$

which is called the complementary slackness condition. This condition is used to determine which constraints are active and which are not. Hence in this case, if the Lagrange multiplier is zero, then the constraint is inactive, whereas if the slack variable is zero the constraint is active. This complementary condition introduces a combinatorial problem that grows exponentially (2^{n_g}) as the number of inequality constraints is increased.

For the optimality conditions, the Lagrange multipliers are required to be non-negative for active constraints, hence putting the stationary conditions together with the non-negative condition, the first-order optimality relations are given as follows [6],

$$\begin{aligned}
&\nabla f + J_h^{\mathsf{T}}\lambda + J_g^{\mathsf{T}}\sigma = 0 \\
&h = 0 \\
&g + s \odot s = 0 \\
&\sigma \odot s = 0 \\
&\sigma \geq 0,
\end{aligned} \tag{2.24}$$

These are known as the Karush-Kuhn-Tucker (KKT) conditions. In addition to these conditions, to make sure that the point found is a minimum, the second order conditions need to be checked (similar to Equation 2.12) and require the Hessian of the Lagrangian, i.e. [6],

$$\begin{aligned}
&p^{\mathsf{T}} H_{\mathcal{L}} p > 0 && \text{for all } p \text{ such that:} \\
&J_h p = 0 && \\
&J_g p \leq 0 && \text{for all active constraints.}
\end{aligned} \tag{2.25}$$

where J_h and J_g are the Jacobian matrices associated with the equality and inequality constraints respectively.

2.5. Penalty methods

The concept of penalty methods is another concept used to convert constrained problems to unconstrained ones by adding a penalty term to the objective function that is associated with the constraints only when the constraints are violated or are about to be violated. Penalty methods can be problematic when directly used with gradient-based optimization, because of difficulties to obtain a solution, but are quite widely used in gradient-free optimization and also for Merit functions (chapter 5) used in line search algorithms [6]. This concept is an important one because it is used in interior point methods which is the choice of gradient-based optimization used for this project. The basic formulation of a penalized function can be given as [6],

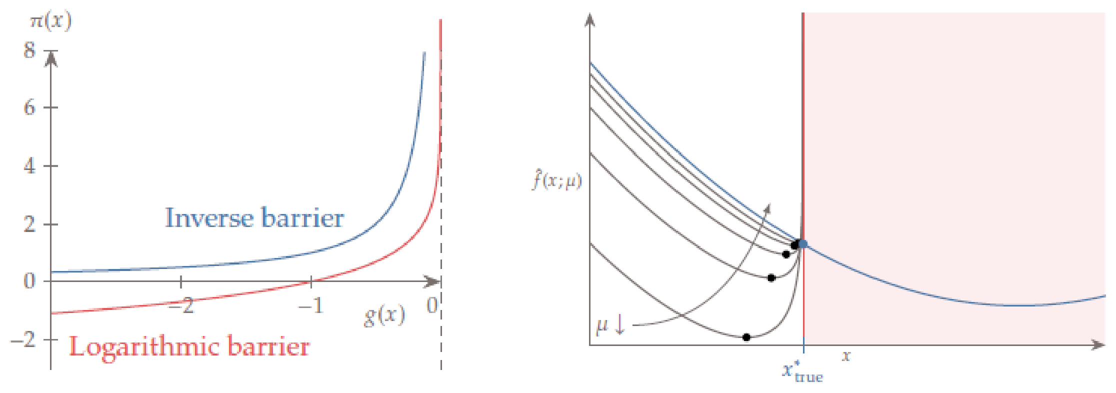
$$\hat{f}(x) = f(x) + \mu\pi(x), \tag{2.26}$$

where $\pi(x)$ is the penalty function and associated with it is the scalar penalty parameter μ. It is similar to the Lagrangian formulation, however, the parameter μ is fixed and not a variable like the Lagrange multipliers.

Penalty parameters can be formulated as interior or exterior penalty methods. Exterior methods, apply penalties if the optimizer violates constraints, while in interior penalty functions, the penalty term is increased as a constraint is approached. The focus of this section is interior penalty methods as it provides the basis for the interior point algorithm covered in chapter 5.

Interior Penalty Methods

Interior Penalty methods similar to the exterior penalty methods are solved as a sequence of unconstrained problems. The penalty parameter is decreased as the iterations progress. The difference here is as stated previously, the interior penalty method looks to maintain feasibility whereas, for exterior penalty methods, the penalty is added only when the constraint is violated. These methods are called the interior point method because as the name suggests, the iterations while searching for the optimum remain within the feasible region. These methods are also known as barrier methods because the penalty



(a) Plot of inverse barrier and Logarithmic barrier penalty functions.

(b) Decreasing μ (black line) approaches the true constrained minimum (blue and red intersection).

Figure 2.10: Barrier function plots [6].

function acts as a barrier to prevent the optimizer from drifting away from the feasible region. For an inequality constraint $g(x) \leq 0$ the Logarithmic barrier function is given as [6],

$$\pi(x) = \sum_{j=1}^{n_q} - \ln\left(-g_j(x)\right), \tag{2.27}$$

This can be substituted into Equation 2.26 to give the penalized objective function as,

$$\hat{f}(x;\mu) = f(x) - \mu \sum_{j=1}^{n_g} \ln(-g_j(x)). \tag{2.28}$$

Another penalty function known as the inverse barrier function is given as,

$$\pi(x) = \sum_{j=1}^{n_g} -\frac{1}{g_j(x)}. \tag{2.29}$$

The plots for the two penalty functions (Equation 2.27 and Equation 2.29) are shown in Figure 2.10a. It can be seen how the Logarithmic barrier is defined over negative values (feasible region) but as soon as $g(x) \to 0$, the barrier term stringently tends to ∞. In Figure 2.10b, the black lines represent the penalized function and the blue lines represent the actual function, while the red region is the infeasible region. It can be seen that as the penalty parameter reduces, the penalized function approaches the true minimum.

One drawback of the method is that it is not defined for infeasible points so the optimizer needs to start optimizing from the feasible region. As it can be seen in Equation 2.29 the equation would yield an undefined value for a positive value, $g(x) > 0$.

2.6. Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is one of the first modern optimization methods. The name stems from the fact that the method solves a sequence of Quadratic Programming (QP) problems to achieve optimality [6]. In this section application of SQP for only equality-constrained problems is covered because the SQP method does not account for inequality constraints as well as the interior-point method (chapter 5) does with respect to whether the constraint is active or inactive (Interior-point optimization is covered in chapter 5).

The KKT conditions as shown in the previous section can be derived by removing the inequality constraints from the problem, and yields the following conditions [6],

$$r = \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ \nabla_\lambda \mathcal{L}(x, \lambda) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + J_h^{\mathsf{T}} \lambda \\ h(x) \end{bmatrix} = 0. \tag{2.30}$$

To solve a sequence of linear systems, Newton's method is applied as follows [6],

$$\begin{aligned} J_r(u_k) p_u &= -r(u_k) \\ u &\equiv \begin{bmatrix} x \\ \lambda \end{bmatrix} \end{aligned} \tag{2.31}$$

Differentiating the vector of residuals given in Equation 2.30 with respect to the vectors in u yields the following block of $n_x + n_h$ equations corresponding to a linear system [6]:

$$\begin{bmatrix} H_{\mathcal{L}} & J_h^T \\ J_h & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -\nabla_x \mathcal{L} \\ -h \end{bmatrix}. \tag{2.32}$$

where $H_{\mathcal{L}}$ is the Hessian of the Lagrangian. These equations can be solved to obtain the descent/search directions, and the design variables along with the Lagrange multipliers are updated at each iteration. The variables do not accept a full Newton step but employ a line search such as the backtracking method given in subsection 2.3.3 [6]:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_x \\ \lambda_{k+1} &= \lambda_k + p_\lambda. \end{aligned} \tag{2.33}$$

where α_k is the selected step size after the line search for the k^th optimization iteration. This method is utilized in the optimization methodology developed for this project for solving an optimization problem with equality constraints.

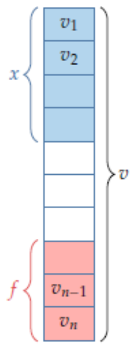
2.7. Automatic Differentiation

In this thesis project, the derivatives used for the gradient-based optimizer explained in chapter 5 are calculated using a method known as Automatic Differentiation which is also known as Algorithmic differentiation (AD). In this section, the basics of AD are covered. AD is a popular approach based on the application of the chain rule across the various steps in a computer program for the evaluation of derivatives of requested expressions [11], [12]. The precision with which the derivatives are computed using AD is determined by the precision with which the function values are evaluated. AD can be formulated in such a way that, the computational cost can either be proportional to the number of variables or the number of functions to be evaluated. more on this will be covered towards the end of this section.

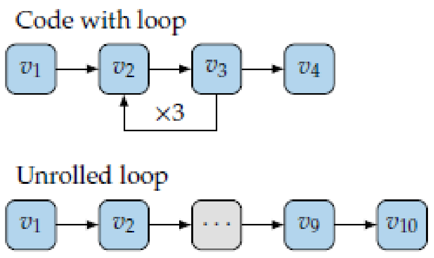
2.7.1. Basic concept

In every computer program or code, there is a sequence of operations that take place in a line-by-line fashion, for instance, addition, multiplication, trigonometric identities, etc. AD can perform symbolic differentiation based on these basic operations and it accumulates the derivatives of each variable using a numerical version of the chain rule. Hence these unary and binary operations over multiple lines can be combined to obtain one elaborate and explicit function expressed over one line of code. The variables in a program can be represented as the sequence $v = [v_1, ..., v_i, ..., v_n]$, where n is the number of variables in the program. It is assumed that one or more of these variables are given at the start of the program, and correspond to x, while towards the end of the sequence, the outputs obtained are f as illustrated in Figure 2.11a. Then another case needs to be considered, where some variables are overwritten in a program due to iterative loops. Hence for AD to work, these variables must not be overwritten but instead stored as different versions as shown in Figure 2.11b. Hence while the sequential accumulation of these lines of code takes place, then a variable assignment corresponding to a particular line of code depends only on variables encountered previously including itself, and is expressed as [6],

$$v_i = v_i(v_1, v_2, \ldots, v_{i-1}). \tag{2.34}$$



(a) All the variables considered by AD with x among the first and f among the last of the variables.



(b) Model to visualize unrolling of loops.

Figure 2.11: Understanding AD [6].

With a sequence of operations and knowing its respective derivatives the numerical equivalent of the chain rule can be applied to obtain the derivatives for an entire sequence. Note, the unrolling of loops shown in Figure 2.11b is just for understanding, and may not be done in actual practice. The chain rule is applied in two ways; the forward mode and the reverse mode. These methods are described below.

Forward-mode AD

In forward-mode AD, an input variable (or seed) is chosen, and the derivatives are obtained by moving forward along the various operations. The chain rule in this case is given as [6],

$$\frac{\mathbf{d}v_i}{\mathbf{d}v_j} = \sum_{k=j}^{i-1} \frac{\partial v_i}{\partial v_k} \frac{\mathbf{d}v_k}{\mathbf{d}v_j} \Rightarrow \dot{v}_i = \sum_{k=j}^{i-1} \frac{\partial v_i}{\partial v_k} \dot{v}_k. \tag{2.35}$$

where k is the iterate, j is the seed point and i is the expression for which the derivative is required.

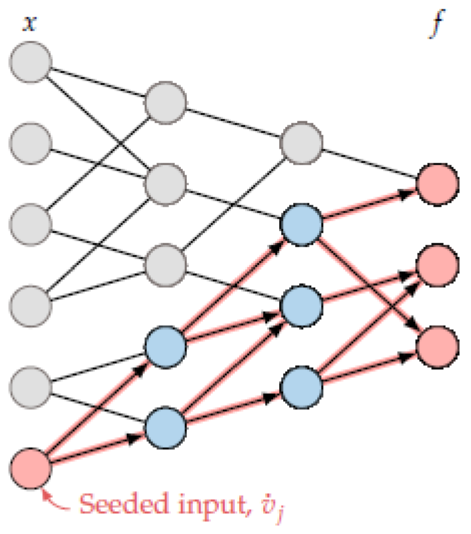


Figure 2.12: Forward-mode propagates derivatives to all the variables that depend on the seeded variable. [6].

The derivatives with respect to the chosen input variable v_j are taken by differentiating symbolically the explicit expression for v_i, which can be computed as the chain rule [6]. A sequence of expressions is evaluated from v_j to $v + i - 1$ as shown in Equation 2.35. The propagation of derivatives from the seed point in forward-mode AD is shown in Figure 2.12.

For instance, consider having four variables v_1, v_1, v_1, and v_4, where $x \equiv v_1$, $f \equiv v_4$, and df/dx is required, then following the chain rule as shown in Equation 2.35 the following is obtained [6]:

$$\begin{aligned}
\dot{v}_1 &= 1 \\
\dot{v}_2 &= \frac{\partial v_2}{\partial v_1}\dot{v}_1 \\
\dot{v}_3 &= \frac{\partial v_3}{\partial v_1}\dot{v}_1 + \frac{\partial v_3}{\partial v_2}\dot{v}_2 \\
\dot{v}_4 &= \frac{\partial v_4}{\partial v_1}\dot{v}_1 + \frac{\partial v_4}{\partial v_2}\dot{v}_2 + \frac{\partial v_4}{\partial v_3}\dot{v}_3 \equiv \frac{\mathbf{d}f}{\mathbf{d}x}.
\end{aligned} \tag{2.36}$$

In the above equation, $\dot{v}_1 = 1$ is in accordance to Equation 2.35. Further, only the partial derivatives of the current operation v_i need to be evaluated, and then multiplied using the total derivatives $\dot{v}$ that have been computed already. This is convenient because only the partial derivatives of the function at hand need to be evaluated. The Jacobian of the variables with respect to themselves for this example is given as [6],

$$J_v = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{\mathrm{d}v_2}{\mathrm{d}v_1} & 1 & 0 & 0 \\ \frac{\mathrm{d}v_3}{\mathrm{d}v_1} & \frac{\mathrm{d}v_3}{\mathrm{d}v_2} & 1 & 0 \\ \frac{\mathrm{d}v_4}{\mathrm{d}v_1} & \frac{\mathrm{d}v_4}{\mathrm{d}v_2} & \frac{\mathrm{d}v_4}{\mathrm{d}v_3} & 1 \end{bmatrix} \tag{2.37}$$

In the Jacobian, it can be seen that with the seed at $j = 1$, the required derivatives are propagated downwards along the diagonal. Hence if the seed had to be set at $j = 2$, then the first row and first column of evaluations would be eliminated with the required Jacobian matrix being a (3×3) instead of a (4×4) matrix. Thus it can be noticed that the cost of forward-mode AD scales linearly with the number of inputs and is independent of the number of outputs.

Reverse-mode AD

In reverse-mode AD, one output variable is chosen and the derivatives are computed backward toward the input until the desired variables are obtained. The chain rule for reverse-mode AD is an alternative form and is given as,

$$\frac{\mathbf{d}v_i}{\mathbf{d}v_j} = \sum_{k=i}^{j+1} \frac{\partial v_k}{\partial v_j} \frac{\mathbf{d}v_i}{\mathbf{d}v_k} \Rightarrow \bar{v}_i = \sum_{k=i}^{j+1} \frac{\partial v_k}{\partial v_j} \bar{v}_k. \tag{2.38}$$

where the summation starts at $k = i$ and is decremented to till $j + 1$ is reached i.e. $j < i$. The index i is fixed at the output of interest with the reverse seed set to $\bar{v}_i = 1$ and then the derivatives are accumulated till the variable index j is reached (in reverse).

Now applying the reverse-mode AD to the same 4 variable examples as shown for forward-mode AD the following sequence of derivative computations can be observed [6],

$$\begin{aligned}
\bar{v}_4 &= 1 \\
\bar{v}_3 &= \frac{\partial v_4}{\partial v_3}\bar{v}_4 \\
\bar{v}_2 &= \frac{\partial v_3}{\partial v_2}\bar{v}_3 + \frac{\partial v_4}{\partial v_2}\bar{v}_4 \\
\bar{v}_1 &= \frac{\partial v_2}{\partial v_1}\bar{v}_2 + \frac{\partial v_3}{\partial v_1}\bar{v}_3 + \frac{\partial v_4}{\partial v_1}\bar{v}_4 \equiv \frac{\mathbf{d}f}{\mathbf{d}x}.
\end{aligned} \tag{2.39}$$

After setting $\bar{v}_4 = 1$ and using the chain rule in reverse as suggested in Equation 2.38, the derivatives $\bar{v}_3$ and so on are evaluated. Not all the variables are dependent on one another, hence a computational graph is created during code evaluation which is used to interpret which variables are dependent on each other and proceed like shown in Figure 2.13 [6].

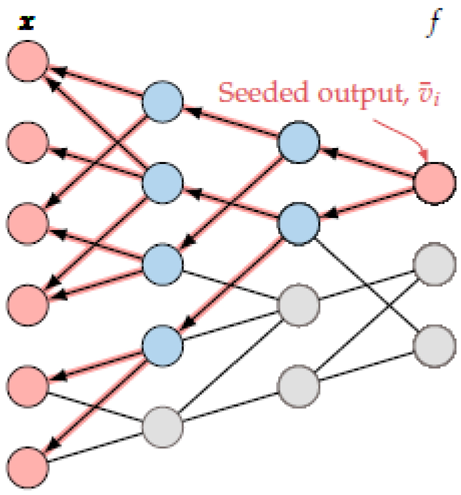


Figure 2.13: Reverse-mode propagates derivatives to all the variables that depend on the seeded variable while moving in reverse. [6].

Once again, the Jacobian matrix of these four variables is the same as given by Equation 2.37 but a difference can be observed. This time the evaluations proceed from the right bottom corner where $\bar{v}_4 = 1$ and the derivatives are propagated upwards along the diagonal, towards the starting variables. If $\bar{v}_3 = 1$ was set as the seed point, then the 4th row and column would disappear making J_v a (3×3) matrix. And as seen in Figure 2.11a, the last few variables are usually functions. Hence it can be concluded that the cost of reverse-mode AD scales linearly with the number of functions being evaluated. A drawback of this method however is that it requires the lines of code to run once in the forward direction to for generating the computational graph to see which variable depends on the other, followed by the calculation of variable values that would be required during the reverse-mode AD for calculating the respective derivatives.

Hence for the selection of the mode for AD based on computational efficiency, the two things that need to be observed by a user are, the number of input variables (n_x) and the number of function evaluations or outputs (n_f). If the number of input variables is larger, the reverse mode AD is the method to choose, whereas if $n_f > n_x$ then using forward-mode AD is beneficial.

Developing a module that can carry out AD for a computer program can be a difficult task, but there is a Python module for AD that is free to use and is known as Autograd. This is a module from the library of Pytorch and a pre-print that presents this work by Paszke et al. is available [13]. The module covers the basic functions required for the optimization algorithm like gradient calculation of a function with respect to a variable, Jacobian calculation, etc and by default runs reverse-mode AD. Hence for this thesis project, this module is implemented for differentiation. However, a drawback of this module is that the data type it utilizes is tensor, and not all other modules in Python accept tensors as input data types, hence making the implementation of AD hectic as the functions associated with these other modules would have to be coded separately to support tensor data type. Hence the creation of an AD module that runs on data types that are supported by other commonly used modules is an interesting topic for a project.

This page was intentionally left blank.

3 The Deflation method

This chapter will cover the basic concept of deflation and the methods applied in optimization. This content serves as the basis for the main contribution of this thesis, which is the deflation constraint (discussed in chapter 6).

Global optimization of non-convex objective functions has been a widely studied topic with many methods that focus on generating multiple minima to find a global minimum or even explore design feasibility. For instance, the optimal design of a water bottle could generate multiple minima that satisfy the design needs (eg. reducing the environmental impact of single-use bottles [14]), however only a few of them would satisfy aesthetic appeal, which is usually a criterion for designs that cannot be quantified as a constraint. The deflation method is one such method that helps to generate multiple distinct local minima and aid in exploring design feasibility.

There are multiple heuristic non-convex optimization techniques with a random restart strategy to scan the design space for multiple local minima [15], [16]. But even with these random restarts, there is no guarantee that the optimizer will not discover solutions or minima that have already been discovered hence increasing computational inefficiency. This issue is remedied by using deflation-based methods. Papadopoulos et al. [17] describe a novel optimization method that combines deflation with barrier methods and primal-dual active set solvers to find multiple stationary points in a non-convex topology optimization problem. Xia et al. [18] present a deflation-based strategy to obtain multiple solutions for the design of thin-walled structures under compression loading that is displacement controlled. Such studies, therefore, demonstrate that the use of deflation methods does enable convergence to distinct solutions despite having to rerun from the same starting point.

3.1. Polynomial deflation

The first instance of the deflation method was first seen in the application for root finding in non-linear polynomial functions [19]. This method was later extended to Partial Differential Equations (PDE) by Farrell et al. [20]. Polynomial deflation however provides an easier visualization of the concept and is covered in this section.

Assume $p(x)$ to be a scalar polynomial function having n multiple roots such as $x_1, x_2, ..., x_n$ and can be found using an iterative method for instance Newton's method [21]. After having evaluated the initial root of $p(x)$, more roots can be found by considering the following deflated function [20],

$$q(x) = \frac{p(x)}{\prod_{i=1}^{n}(x - x_i)}, \tag{3.1}$$

where q(x) is the deflated polynomial function effectively out of which the already obtained roots are removed from the function by the multiplicative term in the denominator of Equation 3.1. This phenomenon can be visualized by considering the simple example demonstrated below.

The polynomial function for this example is considered to be a sine function, $sin(\pi x)$ which of course has infinite bounds and infinite roots, but in this case, it is considered to oscillate between (-3,3).

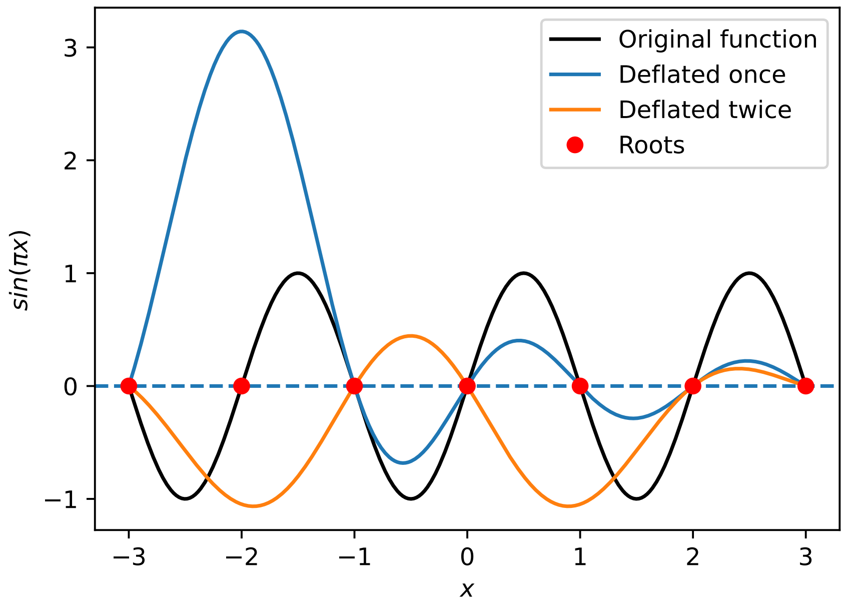


Figure 3.1: Deflated sine function obtained using Equation 3.1

The black colored plot in Figure 3.1 depicts the original sine function while the blue line depicts the once deflated function for the root evaluated at -2. The orange line represents the deflated function out of which two roots have been deflated, which are -2 and 1 (The plots have been generated by coding a python script). The deflated function for the blue and orange plots calculated as Equation 3.1 are respectively given as,

$$\begin{aligned} q(x) &= \frac{sin(\pi x)}{x - (-2)} \\ q(x) &= \frac{sin(\pi x)}{(x - (-2))(x - 1)} \end{aligned} \tag{3.2}$$

Hence in the Figure 3.1 it can be seen how in the plots of the deflated functions, how the roots are completely removed i.e. the function does not cross over zero without affecting the other roots. This essentially provides a very good visual of how deflation works.

3.2. Deflation applied to Nonlinear equations

Farrell et al. cite Farrell2015DeflationEquations worked on extending the concept deflation matrix M and deflation operators m introduced by Brown and Gearhart [22] to implement it for solving Partial Differential Equations (PDE). With this work, they proved that convergence to different solutions from the same starting point was achieved after deflation. The method is explained below.

Consider the residuals for a system of n non-linear equations such that [23],

$$F(x) = 0 \tag{3.3}$$

After solving the system of equations once to get the solution x_1, the deflation matrix and the deflation operator are defined as follows [23],

$$\begin{aligned} M(x; x_1) &= m(x; x_1)\mathcal{I} \\ m(x; x_1) &= ||x - x_1||^{-p} + \sigma \end{aligned} \tag{3.4}$$

where $\mathcal{I}$ is the identity matrix $(n \times n)$, the pole strength or power p dictates the rate at which the function approaches ∞ (the introduced singularity pertaining to m), and the shift parameter σ which ensures that the deflated function gains back original behavior when the normed distance $||x - x_1|| \to \infty$ [18], [23]. The deflated nonlinear system of equations is given as [23],

$$G(x) \equiv M(x; x_1)F(x) = 0 \tag{3.5}$$

This system of equations can be solved to obtain a new solution x_2. Here G satisfies the following properties [18]:

1. The two system of equations, $F(x) = 0$ and $G(x) = 0$ both have the same solutions for all $x \neq x_1$.
2. With the known solution x_1, G will not converge again to x_1 under the assumption $\lim_{x \to x_1} \|G(x)\| > 0$

After having found $\tilde{K}$ multiple solutions, the method in [20] proposes multiplying the deflation matrices and solving the following set of equations [23],

$$\prod_{k=1}^{\tilde{K}} M(x; x_k)F(x) = 0. \tag{3.6}$$

To improve the numerical stability of Equation 3.6, the multiplication can be replaced by summation as claimed by Tarek et al. [23]. Farrell et al. in a recent study [24] have further extended this method for semi-smooth equations as well. But this will not be explained here as the thesis focuses on the optimization of smooth and continuous functions.

Deflation does not guarantee that all the solutions to a problem will be found, however, it provides a powerful tool that complements continuation methods [20]. The deflation method introduced in the methodology of the present thesis is a major contribution of the work, and what has been discussed so far forms a good basis and introduction to understanding how deflation can be applied to optimization problems and is covered in chapter 6.

This page was intentionally left blank.

4 Variable Stiffness Laminates

Laminated composite materials with variable stiffness are those in which the fibers follow a curvilinear path, unlike traditional straight-fiber laminates in which the path of the fibers is constant per layer. Variable-stiffness laminates came into the picture in the late 1980s with the first attempts trying to align the fibers of the laminate in the direction of the principal stresses, to improve characteristics like buckling of composite plates [25], [26]. Biggers and Srinavasan [27] in their work presented a method to construct variable stiffness laminates by stacking layers of constant stiffness laminates over localized regions. In certain manufacturing processes like draping, straight-fibers after undergoing the manufacturing process result in the change in curvature of the fibers, usually associated with single or double curved surfaces [28], [29]. In the following discussion various methods of manufacturing Variable Stiffness (VS) laminates are covered followed by modeling methods which are then compared to choose a method that will be applied in a case study for this thesis.

4.1. Manufacturing of variable stiffness Laminates

With manufacturing processes like Automated fiber placement (AFP), Automated Tape Laying (ATL) and Continuous Tow Shearing (CTS), it is possible to steer the fibers during the deposition of unidirectional composite materials [30], [31]. Even 3D printing is yet another very flexible manufacturing process and currently, some methods have been developed to manufacture composite structures with continuous fiber reinforcements that follow curvilinear paths [32]. Since 3D printing may not seem like a viable option for manufacturing large-scale structures like an aircraft wing, the more matured manufacturing processes i.e. AFP, and CTS are discussed below.

4.1.1. Automated Fiber Placement

The AFP process utilizes a gantry/robotic system with an attached fiber placement head. The AFP head enables multiple strips of composite material, or tows, to be laid over the surface contour accurately as it is dictated by the design. The adhesion between the deposited tows and substrate is ensured by using appropriate process parameters which include heating, compaction, and tensioning systems. A series of tows form a course. Courses are then combined to create a ply, and multiple plies create a laminate [33]. In the previously cited reference, Brasington provides a comprehensive overview of the progression of AFP technology over the course of time.

Like most manufacturing methods, defects can also arise and can be a more prominent factor in VS laminates. For instance, when shifting of the curved path occurs in a certain direction as seen in the first two images of Figure 4.2 a), the shifting direction is the same over the structure, however the axis perpendicular to the course is changing, resulting in gaps or overlaps [34], [35]. Accumulation of overlaps and gaps ultimately results in waviness that can lead to thickness accumulations for overlaps resulting in localized stiffness increase or in the case of gaps a decrease in stiffness. Current software advances do enable the machines to carry out a ”cut-and-restart” procedure but this could result in discontinuities and as illustrated in the work of Chen et al. [36], inevitable gaps and overlaps. Other defects associated with AFP include wrinkling of the material being laid down and twisting of the tows, which can be mitigated with adequate control over process parameters as well. However, methods to model these defects and analyze the structure must also be explored.

Fayazbakhsh et al.[37] suggested a method in their study named the defect layer method where the effects of gaps and overlaps have been taken into account. In this method, a correlation between the volume fraction of defects and elastic properties is found using a micro-mechanical model of the defect, followed by which the properties are assigned to each finite element based on the former correlation. Lastly using FEA, the performance of the laminate is evaluated. Nik et al. [38] also show a method to take into account such defects and an optimization of VS laminates has been carried out. Another interesting study is done by Mishra et al. [39] where they introduced a method that is 45% computationally efficient as compared to [37] but the method captures buckling behavior more on a non-conservative side due to stress concentration not being captured accurately.

Since the thickness values can vary in parts of the laminate due to bending of the tows during the process, ways have been introduced to approximate these thickness variations. Castro et al. in the study of the application of Edge-Based Smoothened Point Interpolation (ES-PIM) for VS laminate buckling have used relations for the variation of thickness. Figure 4.1 displays the parameters of the variable angle tow. First, the effective width of tow relation ($w_e(x, y)$) from Blom et al. [38] was used by Castro et al. and is given as,

$$w_e(x, y) \approx \frac{w_{\text{tow}}}{\sin(\theta(x, y))} \tag{4.1}$$

A smeared thickness h_e is defined in this work, such that the effective volume of the material remains constant despite the change in width and is calculated as,

$$h_e(x) = h_{\text{tow}} \, \frac{w_e(x)}{w_{\text{tow}}} \approx \frac{h_{\text{tow}}}{\sin(\theta(x))} \tag{4.2}$$

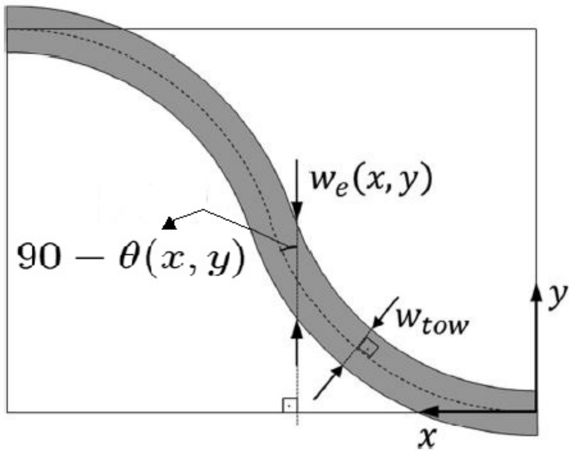


Figure 4.1: Variable angle tow [40]

4.1.2. Continuous Tow Shearing

CTS is a method developed by Kim et al. [31] to provide a novel approach to overcome the various defects induced during AFP such as tow gaps and overlaps by carrying out shear deformation of partially impregnated tows. Hence in this method, the tows are steered through in-plane shear deformation, unlike the AFP method which utilizes bending deformation and ensures that the fibers within the tow can be aligned to have a constant angle. A comparison between AFP and CTS processes is shown in Figure 4.2 and it can be observed that in the AFP process, due to gaps and overlaps, there is local thickening that takes place; whereas in the CTS process, the distribution of thickness is fairly even.

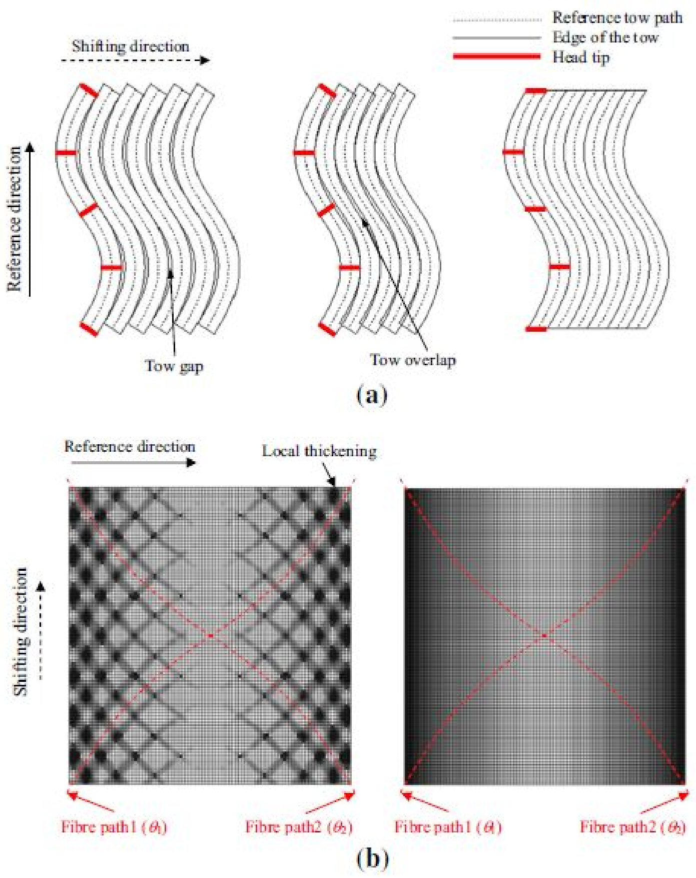


Figure 4.2: Comparison between AFP and CTS: (a) difference of the tow arrangement and head rotation (left and middle: AFP, right: CTS), (b) thickness distribution of a VAT plate with the stacking sequence of $[\theta_1, -\theta_1, -\theta_2, \theta_2]$ (left: AFP, right: CTS) [41]

An important consideration that needs to be made in this process is that there will be variations in the width of the tow and the thickness of the tow (Figure 4.3) as it is being sheared [41]. Kim et al. also suggest ways to parameterize these variations, but first, the parameters involved in the formulation must be understood. Figure 4.4 is a representation of the CTS process using a rectangular element where the reference axis depicts the angle with zero shear which is perpendicular to the shifting direction and the orientation of the CTS head; β is the rotation angle of the reference axis with respect to the x-axis; L is the distance between the intersection points of a line drawn through the element centroid and intersecting with the fiber paths, calculated to compensate for the width adjustment as it is different from the initial width w_0; d is the distance of the intersection point from the centroid of the element. α which is the average fiber angle at the element centroid can be linearly interpolated and is calculated as,

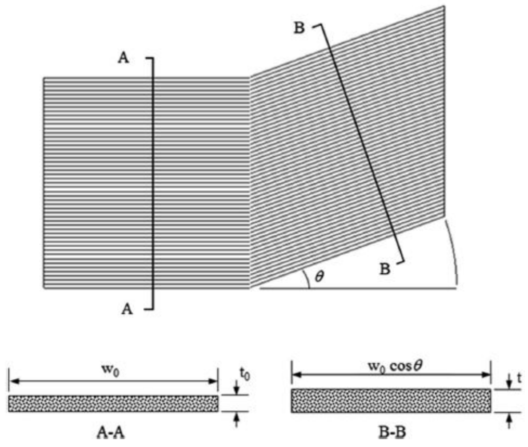


Figure 4.3: Change in thickness and width of tow [41]

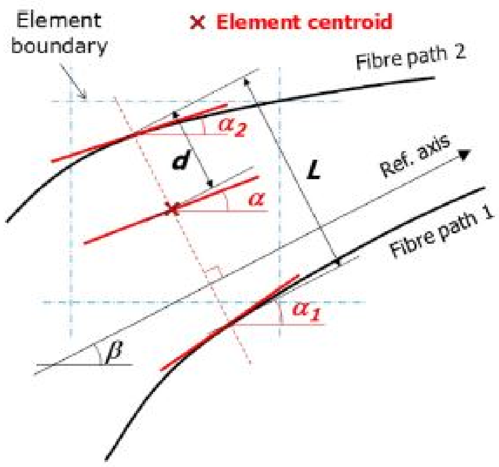


Figure 4.4: Representation of various parameters in CTS with a rectangular element [41]

$$\alpha = \alpha_1 + \frac{d}{L}(\alpha_2 - \alpha_1) \tag{4.3}$$

Kim provides three approaches to calculate the thickness [41],

$$t = t_0 \quad (Op1 : Constant\ thickness) \tag{4.4}$$

$$t = t_0/\cos(\alpha - \beta) \quad (Op2 : Based\ only\ on\ shear\ angle) \tag{4.5}$$

$$t = (w_0/L) \cdot t_0/\cos(\alpha - \beta) \quad (Op3 : Based\ on\ shear\ angle\ and\ tow\ width\ change) \tag{4.6}$$

where t_0 is the original thickness of the tow. These three relations can be used depending on the required parameterization.

4.2. Modeling methods

Variable Stiffness laminates were originally introduced as structures that were tailored for a particular load application resulting in weight reduction. The modeling of these laminates facilitates a better understanding of the loads and opens up opportunities to optimize the structure. Two known methods for modeling VS laminates include: Direct angle parameterization where an interpolation function can be used to define the fiber orientation across each layer of the laminate or by using Lamination Parameters (LP) as primary design variables that define the stiffness properties of the entire laminate and then using those properties to obtain a ply stacking sequence.

4.2.1. Fiber path parameterization

In this method, the fiber orientation angle of each layer of the laminate serves as a design variable and the fiber orientation angles can be defined using many interpolation functions, the simplest one being, linear interpolation. This method was utilized by Gurdal et al. in their work [42] where they studied the in-plane properties and the buckling response of a variable stiffness panel. The rectangular panel is considered to be composed of balanced and symmetric angle ply layers having linear variation of a pair of fiber angles $[\pm\theta(x)]_s$ along any axis (but in this case x). The length of the panel is a and the x-axis is centered along the length of the panel hence spanning from $[\frac{-a}{2}, \frac{a}{2}]$. The expression for the angle is given as,

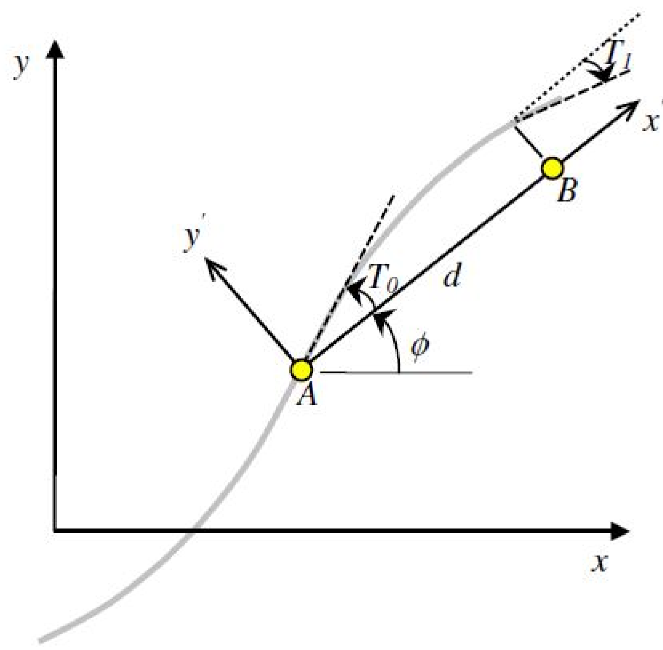


Figure 4.5: Reference path definition of a variable fiber angle layer [42]

$$\theta(x) = \frac{2(T_1 - T_0)}{a}|x| + T_0 \tag{4.7}$$

where T_0 is the fiber angle at the center of the panel and T_1 is the fiber angle at the ends $\pm\frac{a}{2}$. With this simple formulation, closed-form solutions were possible for in-plane stress distribution and displacements of the panels with various boundary conditions by applying uniform edge displacements. The equation 4.7 can be written in a more general sense such that the angle does not need to vary along either x or y axis and can vary along an arbitrary axis x'.

$$\theta(x') = \phi + (T_1 - T_0)\frac{|x'|}{d} + T_0 \tag{4.8}$$

Here ϕ is the angle at which the axis of direction x' is tilted, and d is the characteristic distance between which the fiber angle changes from an angle T_0 to an angle T_1 with respect to the x' direction. This can also be seen in figure 4.5.

A linear interpolation is of course a simple method, but it undermines what can be achieved with variable stiffness laminates. Hence by using higher-order interpolation functions, a more refined variation of the fiber angle can be obtained as observed in a few studies [43], [44].

Paranas et al. [45] represented the fiber angles using Bezier curves while Alhajahmad et al. [46] utilized Lobatto Legendre polynomials to extend the linear fiber angle formulation to a non-linear case. Wu et al. and Guimaraes et al. in their work [43], [44] have utilized an alternative definition for non-linear distribution of fiber angles based on Lagrangian polynomials with the fiber angles varying along both x and y directions. As shown in figure 4.6, the design domain can be considered to consist of $M \times N$ pre-selected reference points in the plate domain. A double summation is utilized to fit the non-linear distribution of the fiber angles and the equation is given as

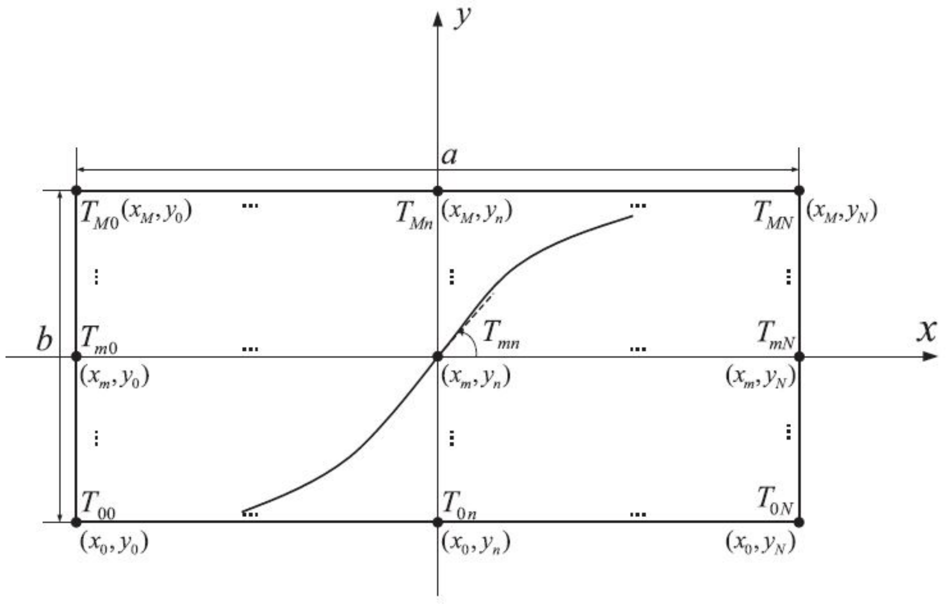


Figure 4.6: Non-linear variation of the fiber orientation with $M \times N$ reference points [44]

$$\theta(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} T_{mn} \cdot \prod_{m \neq i}\left(\frac{x - x_i}{x_m - x_i}\right) \cdot \prod_{n=j}\left(\frac{y - y_j}{y_n - y_j}\right) \tag{4.9}$$

where $(x_i, y_j), (x_m, y_n)$ are the x-y coordinates of the reference points. The advantage of using this formulation is that the coefficient of each term (T_{mn} is also a fiber angle at a specific reference point (x_m, y_n). If the variation occurs only across one direction for instance the x-axis, the equation 4.9 becomes,

$$\theta(x,y) = \sum_{m=0}^{M-1} T_m \cdot \prod_{m \neq i}\left(\frac{x - x_i}{x_m - x_i}\right) \tag{4.10}$$

and similarly, for a certain angle of rotation of the fiber path ϕ, equation 4.10 becomes,

$$\begin{cases} \theta(x,y) = \phi + \theta(x') = \phi + \sum_{m=0}^{M-1} T_m \cdot \prod_{m \neq i}\left(\frac{x' - x_i}{x'_m - x'_i}\right) \\ x = x\cos(\phi) + y\sin(\phi) \end{cases} \tag{4.11}$$

The grid reference points may be chosen randomly and may not be uniform. But, a good selection of reference points would make the optimization process faster and avoid entrapment in local minima. Ghiasi et al. [47] present a great comparison of different methods used for achieving VS laminates and state that the use of a curvilinear function to describe the fiber path and variation of laminate thickness can also reduce the number of design variables, without compromising structural continuity.

Fiber path parameterization indeed seems quite promising to be used for the design of VS laminates, however, a few drawbacks also need to be kept in mind. Firstly the method can prove to be computationally expensive, as with an increasing number of layers the number of design variables would also increase. Secondly, the design variables relating to each other could occupy a highly non-linear and non-convex space, hence making it difficult to find an optimum. Thirdly, to avoid converging on local maxima or minima more of the design space has to be explored to find optimal values in a global sense. Resulting in increased computation.

4.2.2. Lamination Parameters

Lamination parameters were first introduced by Tsai et al. [48], [49] to represent the laminate layup configuration in a compact form. The transformation properties are derived using trigonometric relations (sines and cosines) in terms of multiple angles. The Lamination Parameters are non-dimensional and can in turn be used to obtain the in-plane and bending properties (ABD matrices) as they are related. A great advantage of using lamination parameters is that the number of design variables is reduced and is independent of the number of layers [3], [50]. The 12 variables are given as,

$$\begin{aligned}(V_{1A}, V_{2A}, V_{3A}, V_{4A}) &= \int_{\frac{1}{2}}^{\frac{1}{2}} (\cos 2\theta, \sin 2\theta, \cos 4\theta, \sin 4\theta) dz, \\ (V_{1B}, V_{2B}, V_{3B}, V_{4B}) &= 4\int_{\frac{1}{2}}^{\frac{1}{2}} \bar{z}(\cos 2\theta, \sin 2\theta, \cos 4\theta, \sin 4\theta) d\bar{z}, \\ (V_{1D}, V_{2D}, V_{3D}, V_{4D}) &= 12\int_{-\frac{1}{2}}^{\frac{1}{2}} \bar{z}^2(\cos 2\theta, \sin 2\theta, \cos 4\theta, \sin 4\theta) dz\end{aligned} \tag{4.12}$$

where V_{iA}, V_{iB} and V_{iD} are in-plane, coupling and flexural lamination parameters. $\bar{z}$ is the normalized z coordinate ($\bar{z} = z/h$ through the thickness and θ is the fiber orientation angle of that layer. The [A], [B], and [D] Matrices can be written as a linear function of the lamination parameters and the parameters of Tsai and Pagano as [48],

$$\begin{aligned}A &= h\,(\Gamma_0 + \Gamma_1 V_{1A} + \Gamma_2 V_{2A} + \Gamma_3 V_{3A} + \Gamma_4 V_{4A}) \\ B &= h^2/4\,(\Gamma_1 V_{1B} + \Gamma_2 V_{2B} + \Gamma_3 V_{3B} + \Gamma_4 V_{4B}) \\ D &= h^3/12\,(\Gamma_0 + \Gamma_1 V_{1D} + \Gamma_2 V_{2D} + \Gamma_3 V_{3D} + \Gamma_4 V_{4D})\end{aligned} \tag{4.13}$$

where the material invariant matrices Γ_i's are given by the parameters of Tsai and Pagano which are material invariants. These elements can be derived as follows [51],

$$\begin{aligned}U_1 &= (3Q_{11} + 3Q_{22} + 2Q_{12} + 4Q_{66})/8, \\ U_2 &= (Q_{11} - Q_{22})/2, \\ U_3 &= (Q_{11} + Q_{22} - 2Q_{12} - 4Q_{66})/8, \\ U_4 &= (Q_{11} + Q_{22} + 6Q_{12} - 4Q_{66})/8, \\ U_5 &= (Q_{11} + Q_{22} - 2Q_{12} + 4Q_{66})/8.\end{aligned} \tag{4.14}$$

where Q_ij are the elements of the reduced stiffness matrix and are given as,

$$\begin{aligned}Q_{11} &= E_1/(1 - \nu_{12}\nu_{21}), \\ Q_{22} &= E_2/(1 - \nu_{12}\nu_{21}), \\ Q_{12} &= \nu_{12}E_2/(1 - \nu_{12}\nu_{21}) = \nu_{21}E_1/(1 - \nu_{12}\nu_{21}), \\ Q_{16} &= G_{12}.\end{aligned} \tag{4.15}$$

In this equation, E is the elastic modulus, ν is the Poisson ratio and G is the shear modulus. Now based on the material invariants U the material invariant matrices can be written as,

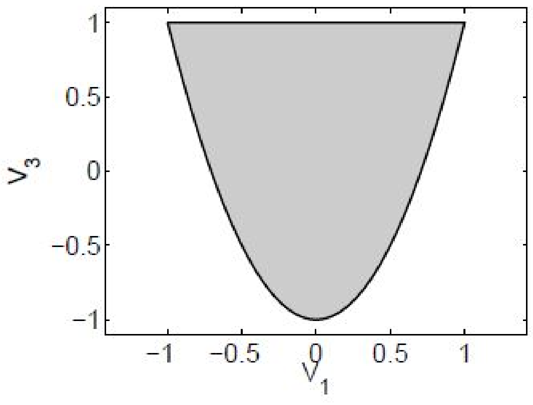


Figure 4.7: Feasible domain for either the in-plane LPs V_{1A} and V_{3A} or only the flexural LPs V_{1D} and V_{3D} [55]

$$\Gamma_0 = \begin{bmatrix} U_1 & U_4 & 0 \\ U_4 & U_1 & 0 \\ 0 & 0 & U_5 \end{bmatrix}, \qquad \Gamma_1 = \begin{bmatrix} U_2 & 0 & 0 \\ 0 & -U_2 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$
$$\Gamma_2 = \begin{bmatrix} 0 & 0 & U_2/2 \\ 0 & 0 & U_2/2 \\ U_2/2 & U_2/2 & 0 \end{bmatrix}, \quad \Gamma_3 = \begin{bmatrix} U_3 & -U_3 & 0 \\ -U_3 & U_3 & 0 \\ 0 & 0 & -U_3 \end{bmatrix}, \tag{4.16}$$
$$\Gamma_4 = \begin{bmatrix} 0 & 0 & U_3 \\ 0 & 0 & U_3 \\ U_3 & -U_3 & 0 \end{bmatrix}$$
.

The linear dependence of the [ABD] matrix on the Lamination parameters as seen in Equation 4.13 seems to be quite useful as proved in the works of Grenestedt et al., that the feasible region of the lamination parameters is indeed convex in nature [52]. However, a difficulty that has been faced is, developing an explicit relation for relating the 12 Lamination parameters. Miki et al. [53], [54] in their works defined the feasible region describing either two in-plane parameters or two bending parameters to characterize the stiffness of an orthotropic plate as,

$$\begin{gathered} V_3 \geqslant 2V_1^2 - 1, \\ -1 \leqslant V_i \leqslant 1 (i = 1, 3) \end{gathered} \tag{4.17}$$

the feasible regions obtained using these relations are shown in the work of van Campen [55] and is depicted in the figure 4.7.

Later, two dimensional projections of the feasible region were explored by Grenestedt [52] using 4 lamination parameters. In the work of Fukunaga et al., [56], the feasible region of four in-plane or flexural lamination parameters was depicted using the following relation,

$$\begin{aligned} &2V_1^2(1 - V_3) + 2V_2^2(1 + V_3) + V_3^2 + V_4^2 - 4V_1V_2V_4 \leqslant 1 \\ &V_1^2 + V_2^2 \leqslant 1 \\ &-1 \leqslant V_3 \leqslant 1 \end{aligned} \tag{4.18}$$

An explicit relation for the feasible region combining four coupled lamination parameters for orthotropic laminates has been derived by Wu et al. [57] by using the Schwarz inequality along with algebraic identities to relate in-plane, coupling, and flexural lamination parameters. The relations are given as follows,

$$\begin{aligned}
&5\left(V_{1A}-V_{1D}\right)^2-2\left(1+V_{3A}-2V_{1A}^2\right)\leqslant 0\\
&\left(V_{3A}-4tV_{1A}+1+2t^2\right)^3-4\left(1+2|t|+t^2\right)^2\left(V_{3D}-4tV_{1D}+1+2t^2\right)\leqslant 0\\
&\left(4tV_{1A}-V_{3A}+1+4|t|\right)^3-4\left(1+2|t|+t^2\right)^2\left(4tV_{1D}-V_{3D}+1+4|t|\right)\leqslant 0
\end{aligned} \tag{4.19}$$

where $t \in [-1, 1]$ and in this paper when t=[-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1], the results are quite satisfactory. The feasible region plots can be viewed in Figure A.3 [57].

Modeling Technique	**Benefits**	**Drawbacks**
Fiber Path Parametrization	• Few number of design variables • Continuous smooth path • Function definition eases curvature constraint application	• Limited design space by path function • Highly non-linear and non-convex • Requires function definitions for different developable surfaces
Direct Stiffness Modeling (Lamination parameters)	• Entire design space encompassed • Set of continuous design variables defined by a convex region • Number of design variables independent of number of layers	• Multi-level optimization required for post-processing fiber angles and fiber path • Feasible region required for interrelating stiffness properties

Table 4.1: Benefits and drawbacks of the two VS laminate modeling techniques [3]

After having seen the few advantages of using lamination parameters for designing Variable Stiffness laminates, moving to the drawbacks. The LP parameterization method assumes all plies to be of the same material. Next, After finding the distribution of the lamination parameters across the design domain, the fiber orientation angles also need to be determined and this cannot be done empirically, hence the results may not be unique. Multiple research papers display methods to obtain fiber orientation angles from the optimized values of lamination parameters. Van Campen et al. [2] utilized a gradient descent approach to find the optimal lamination parameters followed by the usage of Genetic algorithm (GA) treating the steering constraint by a penalty approach and provide starting points for the gradient-based optimizer where the minimum loss of performance was achieved in the fiber angle retrieval process. In this work, they have made it possible to include manufacturing constraints in the design process using cellular automation (CA). For more information on CA, readers can refer to this book [58]. Setoodeh et al. [50] however took a different approach to finding the fiber orientation angles. Firstly the optimal distribution of the lamination parameters was obtained and then curve fitting techniques were used to obtain a continuous fiber orientation angle distribution that would in turn be evaluated to find the lamination parameters and see how close it is to the optimal values. The fiber angles are expanded using a set of basis functions and unknown coefficients. these unknown coefficients are then computed such that the assumed form corresponds to the optimal distribution of the lamination parameters in a least square sense. So as given by van Campen [55] and also Setoodeh et al. [50] the most straightforward objective would be the least square distance to the Lamination parameter's optimum value, which can be written as,

$$f = |V - V^*| \text{ with } V = \{V_{1A}, V_{3A}, V_{1D}, V_{3D}\} \tag{4.20}$$

V^* indicates the optimum lamination parameters. This objective function can be used in the thesis to provide a good case study for fiber angle retrieval using lamination parameters.

A summary of the various benefits and drawbacks of fiber path parameterization and modeling with lamination parameters can be seen in Table 4.1 obtained from [3]. In addition to what has been discussed so far, it is important to note that the thickness of the tows is assumed to be constant in the calculations associated with Lamination parameters which as we have seen from the manufacturing section (section 4.1 is not the case. However, the relations discussed for the variation of thickness have been applied in works that utilize fiber path parameterization [40], [59]. Hence fiber path parameterization seems to be the more appealing solution compared to the other, but owing to the advantages of

Lamination parameters it would be worthwhile to explore if the thickness variations can be implemented in this method as well. Hence it can be possibly pursued in future studies do try to do so.

Part II

Methodology

5 Gradient-based Optimization Algorithm

The literature review provided in chapter 2 gives an overview of how gradient-based optimization works and the different aspects of the method. In this chapter, the interior-point optimization algorithm utilized for this thesis will be detailed and the effect of the main parameters affecting this optimizer will be explained by means of test cases. The optimization algorithm is programmed using the Python programming language.

5.1. Interior - Point Method

The name interior-point stems from interior penalty methods that apply penalization to the constraints in the optimization problem such that a penalization term is associated with the constraints and optimization occurs within the constrained domain. The penalty term increases as the optimizer moves toward the boundary of the constrained domain.

In the method of interior-point optimization, the key difference as compared to interior penalty methods is that the constraints are not directly penalized, but instead the penalization acts on slack variables that are added to the constraints. The formulation is written as follows [6],

$$\begin{aligned} \underset{x,s}{\text{minimize}} \quad & f(x) - \mu_b \textstyle\sum_{j=1}^{n_g} \ln s_j \\ \text{subject to} \quad & h(x) = 0 \\ & g(x) + s = 0 \end{aligned} \tag{5.1}$$

where mu_b is the barrier parameter. The resulting formulation turns the inequality constraint into an equality constraint with the addition of the slack variables s. Newton's method can be applied to solve the KKT system of equation for the above formulation in Equation 5.1, where it can be noted that the logarithm barrier term (refer chapter 2) associated with the slack variables is only defined for positive s values and would act as a barrier for negative s values. Owing to the positive s values, $g(x^*) < 0$ at the solution, hence satisfying the inequality constraints [6].

The interior-point formulation (Equation 5.1) is equivalent to the original constrained problem within the limit; $\mu_b \rightarrow 0$. Hence, a sequence of solutions needs to be obtained to the problem such that nu_b tends to zero.

As explained in chapter 2, a constrained problem can be reformulated to be an unconstrained problem by utilizing the Lagrangian function along with associated Lagrange variables. The Lagrangian for this problem is given as [6],

$$\mathcal{L}(x, \lambda, \sigma, s) = f(x) + \mu_b e^\top \ln s + h(x)^\top \lambda + (g(x) + s)^\top \sigma, \tag{5.2}$$

Here, ln$s$ is a n_g vector with its components being the logarithms of each value of s, $e = [1, ..., 1]$ is a n_g vector of 1s which is introduced to express the sum in vector form, λ is the Lagrange variable vector associated with the equality constraints and σ is the Lagrange variable associated with the inequality constraints. The KKT conditions can be derived by taking the derivatives with respect to x, λ, σ and s, which is given as,

$$\begin{aligned} &\nabla f(x) + J_h(x)^\top \lambda + J_g(x)^\top \sigma = 0 \\ &h = 0 \\ &g + s = 0 \\ &-\mu_b e + S\sigma = 0, \end{aligned} \tag{5.3}$$

where S is a diagonal matrix with its diagonal values given by the slack variable vector. With this set of residual equations, Newton's method can be applied. Taking the Jacobian of the equations in Equation 5.3, the following linear system is obtained:

$$\begin{bmatrix} H_L(x) & J_h(x)^\top & J_g(x)^\top & 0 \\ J_h(x) & 0 & 0 & 0 \\ J_g(x) & 0 & 0 & I \\ 0 & 0 & S & \Sigma \end{bmatrix} \begin{bmatrix} p_x \\ p_\lambda \\ p_\sigma \\ p_s \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda, \sigma) \\ h(x) \\ g(x) + s \\ S\sigma - \mu_b e \end{bmatrix} \tag{5.4}$$

where Σ is a diagonal matrix whose entries are given by the values of σ vector, and I is the identity matrix. For numerical efficiency, the system can be made symmetric by multiplying the last equation in Equation 5.4 with S^{-1}, which can be calculated as $S^{-1}_{kk} = 1/s_k$. This results in the following symmetric linear system:

$$\begin{bmatrix} H_L(x) & J_h(x)^\top & J_g(x)^\top & 0 \\ J_h(x) & 0 & 0 & 0 \\ J_g(x) & 0 & 0 & I \\ 0 & 0 & I & S^{-1}\Sigma \end{bmatrix} \begin{bmatrix} p_x \\ p_\lambda \\ p_\sigma \\ p_s \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda, \sigma) \\ h(x) \\ g(x) + s \\ \sigma - \mu_b S^{-1} e \end{bmatrix} \tag{5.5}$$

This linear system of equations (Equation 5.5) can be now solved using a matrix equation solver module associated with a programming language of choice. The Figure 5.1 depicts the structure and block sizes of what makes up the interior-point system as given in Equation 5.5.

5.1.1. Merit function

In the book of Martins and Ning [6], merit functions have been discussed and are something that has been implemented into this optimizer. In constrained optimization, the decrease in objective function often competes with the feasibility of a newfound optimum. Especially during a line search, the newfound point can either result in a decrease in the objective function and increased infeasibility, or an increase in the objective function and decreased infeasibility. Hence a way to include these two metrics in a line search needs to be implemented. The Lagrangian function does take into account the two metrics, however, in every iteration, the values of the Lagrange multipliers are just estimates and may be inaccurate. Merit functions provide a more elegant way of combining the two metrics. The following merit function is used in the line-search algorithm used for this project:

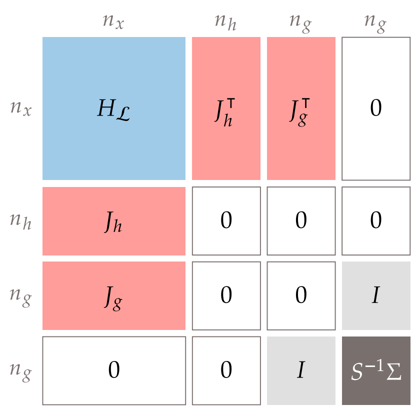


Figure 5.1: Shape and structure of the interior-point system matrix given in Equation 5.5 [6]

$$\hat{f}(x) = f(x) - \mu_b \sum_{i=1}^{n_g} \ln s_i + \frac{1}{2}\mu_p \left(\|h(x)\|^2 + \|g(x) + s\|^2\right), \tag{5.6}$$

where μ_b is the barrier parameter from Equation 5.1, and μ_p is the penalty parameter. The downside of using such a merit function is that it gets quite difficult to choose a suitable value for the penalty parameter ν_p. The parameter needs to be large enough to ensure the feasibility of the newfound point, however, if it is too large, it will result in a smaller step size. This would result in a higher number of iterations to reach convergence. For the case studies explored in this thesis, the value of this parameter varied between 4 and 10.

5.1.2. Maximum allowed step size

The maximum allowed value for the step size can be calculated before the line search because the value of s and p_s is known and it is required that,

$$s + \alpha_{max} p_s = \tau s, \tag{5.7}$$

where τ is a small value (for instance, $\tau = 0.005$). The maximum step (α_{max}) is therefore the smallest positive value that satisfies the equation for all entries in s. If this is not followed, the resulting s values would be negative, hence triggering the Logarithmic barrier term shown in Equation 5.1 The algorithm used for determining the maximum feasible step size is given below in Algorithm 2.

After obtaining this maximum feasible step, a simple backtracking line search can be implemented as shown in subsection 2.3.3 to obtain a new α_k corresponding to the k^{th} iteration. The design variables and slack variables can be updated as follows,

$$\begin{aligned} &x_{k+1} = x_k + \alpha_k p_x, \; where \; \alpha_k \in (0, \alpha_{max}] \\ &s_{k+1} = s_k + \alpha_k p_s \end{aligned} \tag{5.8}$$

Following this, the step size (α_σ) for the Lagrange multipliers (σ) also needs to be calculated such that they maintain positive values. The procedure depicted in Algorithm 2 can be used to do so. The two sets of Lagrange multipliers are then updated as follows,

Algorithm 2 Maximum feasible step size

```
Inputs:
 $s$                                              ▷ Current slack values
 $p_s$                                            ▷ Proposed step
 $\tau$                                           ▷ Fractional tolerance(around 0.005

Output:
 $\alpha_{max}$                                   ▷ Maximum feasible step length

$\alpha_{max} = 1$
for i = 1 to $n_g$ do
    $\alpha = (\tau - 1)\frac{s_i}{p_{s_i}}$
    if $\alpha > 0$ then
        $\alpha_{max} = min(\alpha_{max}, \alpha)$
    end if
end for
```

$$\begin{aligned} \lambda_{k+1} &= \lambda_k + \alpha_k p_\lambda, \\ \sigma_{k+1} &= \sigma_k + \alpha_k p_\sigma \end{aligned} \tag{5.9}$$

5.1.3. Barrier parameter update

The barrier parameter is the parameter (μ_b) associated with the Logarithm term in Equation 5.1, and a very simple approach is to update it using a multiplicative factor, such as, [6],

$$\mu_{b_{k+1}} = \rho \mu_{b_k} \tag{5.10}$$

where the typical value for ρ is around 0.2. However, in this thesis, a more adaptive method proposed by Wachter et al.[60] is utilized, with the expression given as,

$$\mu_{b_{k+1}} = \max\left\{\frac{\epsilon_{tol}}{10}, \min\left\{\kappa_\mu \mu_{b_k}, \mu_{b_k}^{\theta_\mu}\right\}\right\} \tag{5.11}$$

where the constants $\kappa_\mu \in (0,1)$ and $\theta_\mu \in (1,2)$. This way, the barrier term reduces at a superlinear rate and, as it is known, the barrier problem tends to the original objective when $\mu_b \to 0$. A decrease in the barrier term tending to zero, however, could result in numerical difficulties as the iterations progress, which can be avoided by defining a tolerance (ϵ_{tol} such that μ_b does not drop below this tolerance value.

5.2. Developing the Optimization Algorithm

After covering the fundamentals of the interior-point method in the previous section, the optimization algorithm is explained in this section. To develop an optimizer that is capable of handling versatile inputs, i.e. constrained and unconstrained optimization problems, the possible type of problems are narrowed down to four types:

1. Firstly, a fully unconstrained optimization problem which can be solved by applying Newton's method to the simple unconstrained conditions of optimality as discussed in chapter 2.
2. Secondly, an optimization problem with only equality constraints that can be solved by employing the method of SQP which is discussed in chapter 2

3. Thirdly, an optimization problem with only inequality constraints which can be solved using the interior-point method explained above.
4. Lastly, an optimization problem with both equality, and inequality constraints which can again be solved using the interior point method.

The algorithm for optimization used in this thesis is given below. The algorithm has been divided into two parts shown in Algorithm 3 and Algorithm 4 just for the purpose of illustration. The algorithm detects the kind of optimization problem based on the size of the constraint vectors that serve as inputs to the function. In Algorithm 3 (Part: one), the first type is depicted where the vectors for the inequality constraint, as well as equality constraints, is empty, and hence it is an unconstrained problem. The Quasi-Newton approach is applied here, and in the rest of the types, to update the Hessian of the objective. The Newton's method utilizes second-order information which results in better search directions, but calculating the Hessian proves to be a task that may be computationally expensive. The idea behind the Quasi-Newton method is to use the gradients (first-order information) during each iteration to develop an approximation of the Hessian. The book of Martins and Ning [6] provides a great explanation of how the approximation is derived and results in the following expression for the approximation,

$$\tilde{H}_{\mathcal{L}_{k+1}} = \tilde{H}_{\mathcal{L}_k} - \frac{\tilde{H}_{\mathcal{L}_k} s_k s_k^{\mathrm{T}} \tilde{H}_{\mathcal{L}_k}}{s_k^{\mathrm{T}} \tilde{H}_{\mathcal{L}_k} s_k} + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k} \,, \tag{5.12}$$

where,

$$\begin{aligned} s_k &= x_{k+1} - x_k \\ y_k &= \nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1}) \,. \end{aligned} \tag{5.13}$$

note that s_k is the step in the design variable space that resulted from the last line search. The value of the Lagrange multipliers is set to the latest obtained values while approximating the curvature of the Lagrangian, because only the curvature in the design variable space is required. However, in the unconstrained case, there are no Lagrange multipliers and the curvature is directly associated with the objective function. Hence, the calculations are done with respect to the objective function $f(x)$ instead of the Lagrangian $\mathcal{L}$.

In Algorithm 3 it can be seen that the second 'if' statement evaluates the size of the constraint vectors once again and requires that the inequality constraint vector be empty while the equality constraint to not be empty. Additionally, now the Lagrange multipliers associated with these constraints also come into play and hence need to be taken into account in the Lagrangian.

In Algorithm 4 (Part: Two) the third case, i.e. optimization problem with inequality constraint is handled. The first 'if' statement in this algorithm requires the vector of inequality constraints to not be empty while the vector of equality constraints must be empty. Under this part of the algorithm, the notable difference compared to the previous one is the inclusion of the slack variables associated with the inequality constraints. Now the barrier term also needs to be included as discussed in section 5.1.

Lastly, in the same algorithm (Algorithm 4), the KKT system discussed in section 5.1 is solved with both equality and inequality constraints using the interior-point method.

Algorithm 3 Optimization Algorithm (Part: One)

```
Inputs:
  $x_0$                                               ▷ Starting point
  $\tau_{opt}$                                        ▷ Optimality tolerance (around $10^{-5}$)
  $\tau_{convg}$                                      ▷ Convergence tolerance (eg. $10^{-5}$)

Outputs:
  $x^*$                                               ▷ Optimal point
  $f(x^*)$                                            ▷ Optimal function value

convergence = 10                    ▷ Initialize the convergence value higher than $\tau_{convg}$

if h(x) and g(x) are empty vectors then                     ▷ Unconstrained Optimization
    $\tilde{H}_{f_0} = I$               ▷ Initialize Hessian of Lagrangian approximation to identity matrix
    k = 0
    while $\|\nabla_x f\| \infty > \tau_{opt}$ or convergence $> \tau_{convg}$ do
        Evaluate $\nabla_x f$
        Solve the Quasi-Newton system of equations for $p_x$:
        $\left[\tilde{H}_{f_k}\right] [p_x] = -\left[\nabla_x f(x)\right]$
        $\alpha_k$ = backtrack$(p_x, 1)$                      ▷ Line search using Algorithm 1
        $x_{k+1} = x_k + \alpha_k p_x$                        ▷ Update design variables
        Update $\tilde{H}_{f_{k+1}}$          ▷ Compute quasi-newton approximation using Equation 5.12
        $k = k + 1$                                          ▷ Update iteration value
        Calculate convergence value second iteration on-wards.
    end while
end if

if $h(x)$ is not an empty vector and $g(x)$ is empty then     ▷ Only equality constraints
    $\tilde{H}_{\mathcal{L}_0} = I$     ▷ Initialize Hessian of Lagrangian approximation to identity matrix
    $k = 0$                                                  ▷ Initialize number of iterations
    $\lambda_0 = 0$                                          ▷ Initialize Lagrange multipliers
    while $\|\nabla_x \mathcal{L}\| \infty > \tau_{opt}$ or convergence $> \tau_{convg}$ do
        Evaluate $\nabla_x \mathcal{L}$
        Solve the KKT system for $p$:
        $\begin{bmatrix} \tilde{H}_{L_k} & J_h(x)^\top \\ J_h(x) & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_\lambda \end{bmatrix} = -\begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ h(x) \end{bmatrix}$
        $\alpha_k$ = backtrack$(p_x, 1)$                      ▷ Line search using Algorithm 1
        $\alpha_\lambda$ = alphamax$(\lambda, p_\lambda)$     ▷ Use Algorithm 2
        $x_{k+1} = x_k + \alpha_k p_x$                        ▷ Update design variables
        $\lambda_{k+1} = \lambda_k + \alpha_\lambda p_\lambda$  ▷ Update equality Lagrange multipliers
        Update $\tilde{H}_{\mathcal{L}_{k+1}}$  ▷ Compute quasi-newton approximation using Equation 5.12
        $k = k + 1$                                          ▷ Update iteration value
        Calculate convergence value second iteration on-wards.
    end while
end if
```

Algorithm 4 Optimization Algorithm contd. (Part: Two)

```
if $h(x)$ is an empty vector and $g(x)$ is not empty then    ▷ Only inequality constraints
    $\tilde{H}_{\mathcal{L}_0} = I$    ▷ Initialize Hessian of Lagrangian approximation to identity matrix
    $k = 0$    ▷ Initialize number of iterations
    $\sigma_0 = 0$    ▷ Initialize Lagrange multipliers
    $s_0 = 1$    ▷ Initial slack variables
    while $\|\nabla_x \mathcal{L}\| \infty > \tau_{opt}$ or convergence $> \tau_{convg}$ do
        Evaluate $\nabla_x \mathcal{L}$
        Solve the KKT system for $p$:
        $\begin{bmatrix} \tilde{H}_{\mathcal{L}_k} & J_g(x)^\top & 0 \\ J_g(x) & 0 & I \\ 0 & I & S^{-1}\Sigma \end{bmatrix} \begin{bmatrix} p_x \\ p_\sigma \\ p_s \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda, \sigma) \\ g(x) + s \\ \sigma - \mu_b S^{-1} e \end{bmatrix}$
        $\alpha_{max} = \text{alphamax}(s, p_s)$    ▷ Use Algorithm 2
        $\alpha_k = \text{backtrack}(p_x, p_s, \alpha_{max})$    ▷ Line search using Algorithm 1
        $\alpha_\sigma = \text{alphamax}(\sigma, p_\sigma)$    ▷ Use Algorithm 2
        $x_{k+1} = x_k + \alpha_k p_x$    ▷ Update design variables
        $s_{k+1} = s_k + \alpha_k p_s$    ▷ Update design slack variables
        $\sigma_{k+1} = \sigma_k + \alpha_\sigma p_\sigma$    ▷ Update inequality Lagrange multipliers
        Update $\tilde{H}_{\mathcal{L}_{k+1}}$    ▷ Compute quasi-newton approximation using Equation 5.12
        Update $\mu_b$    ▷ Compute as shown in Equation 5.11
        $k = k + 1$    ▷ Update iteration value
        Compute convergence value second iteration on-wards.
    end while
end if

if both $h(x)$ and $g(x)$ are not empty vectors then    ▷ Both, equality and inequality constraints
    $\tilde{H}_{\mathcal{L}_0} = I$    ▷ Initialize Hessian of Lagrangian approximation to identity matrix
    $k = 0$    ▷ Initialize number of iterations
    $\sigma_0 = 0$    ▷ Initialize Lagrange multipliers
    $s_0 = 1$    ▷ Initial slack variables
    while $\|\nabla_x \mathcal{L}\| \infty > \tau_{opt}$ or convergence $> \tau_{convg}$ do
        Evaluate $\nabla_x \mathcal{L}$
        Solve the KKT system for $p$:
        $\begin{bmatrix} \tilde{H}_{\mathcal{L}_k} & J_h(x)^\top & J_g(x)^\top & 0 \\ J_h(x) & 0 & 0 & 0 \\ J_g(x) & 0 & 0 & I \\ 0 & 0 & I & S^{-1}\Sigma \end{bmatrix} \begin{bmatrix} p_x \\ p_\lambda \\ p_\sigma \\ p_s \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda, \sigma) \\ h(x) \\ g(x) + s \\ \sigma - \mu_b S^{-1} e \end{bmatrix}$
        $\alpha_{max} = \text{alphamax}(s, p_s)$    ▷ Use Algorithm 2
        $\alpha_k = \text{backtrack}(p_x, p_s, \alpha_{max})$    ▷ Line search using Algorithm 1
        $\alpha_\sigma = \text{alphamax}(\sigma, p_\sigma)$    ▷ Use Algorithm 2
        $x_{k+1} = x_k + \alpha_k p_x$    ▷ Update design variables
        $s_{k+1} = s_k + \alpha_k p_s$    ▷ Update design slack variables
        $\sigma_{k+1} = \sigma_k + \alpha_\sigma p_\sigma$    ▷ Update inequality Lagrange multipliers
        $\lambda_{k+1} = \lambda_k + \alpha_\sigma p_\lambda$    ▷ Update equality Lagrange multipliers
        Update $\tilde{H}_{\mathcal{L}_{k+1}}$    ▷ Compute quasi-newton approximation using Equation 5.12
        Update $\mu_b$    ▷ Compute as shown in Equation 5.11
        $k = k + 1$    ▷ Update iteration value
        Compute convergence value second iteration on-wards.
    end while
end if
```

5.3. Testing the Optimization Algorithm

The next step after having developed the optimization Algorithm is to test if it works, and what are the effects of varying its parameters. In this section, a few basic optimization problems using simple functions and also problems that have been discussed in the book of Martins and Ning [6] are used as test cases for the Optimization Algorithm.

5.3.1. Unconstrained Optimization

In the previous section, the requirements for the unconstrained optimization were covered in the Algorithm. Now to test if the algorithm can find the minimum of a simple, two-variable, unconstrained problem is implemented. The problem is defined as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{where} \quad & f(x) = {x_1}^2 + 5{x_2}^2 \end{aligned} \tag{5.14}$$

This problem can be visualized using a 3D plot as shown in Figure 5.2. It can be seen that owing to the parabolic nature of the variables, the problem is indeed a convex optimization problem, resulting in a single global optima.

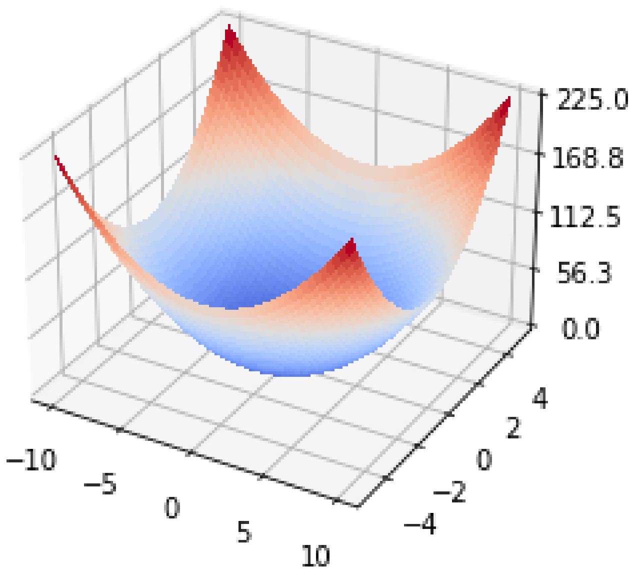


Figure 5.2: Visualization of 2-variable problem (Equation 5.14) using a 3D plot.

This problem can also be visualized on the 2D plane with the use of contour plots. In the Figure 5.3, the path taken by the optimizer starting from the point x_0 to the optimal point (global minimum) x^* is highlighted in green. Since this is an unconstrained problem the only changes that can be made is in the line search algorithm where the constant associated with the sufficient decrease condition (μ_1) and also the constant (ρ) for decreasing the step-size (α) can be varied (Line search is explained in subsection 2.3.3). With each different problem, varying these parameters would result in a change of the path taken by the optimizer, as well as the number of iterations. Hence, only by trying different values, one can determine the ideal parameter values for a particular problem. In this case, $\rho = 0.5$ and $\mu_1 = 0.0001$ were used.

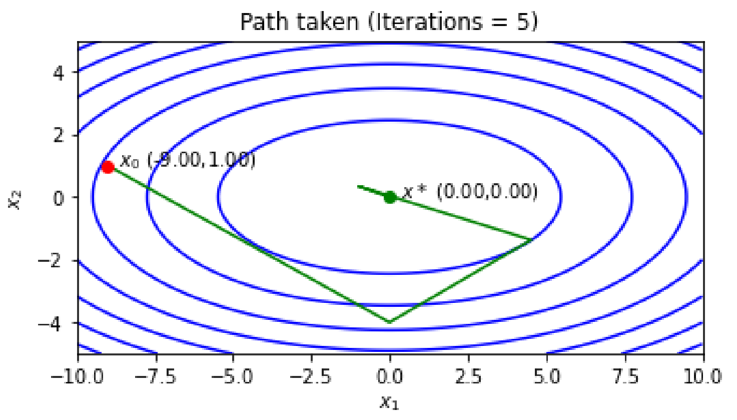


Figure 5.3: Path taken by the optimizer in a 2-variable problem (Equation 5.14).

5.3.2. Equality constraint problem

For this test case, a problem given in the book of Martins and Ning [6] is implemented. The optimization problem is given as follows,

$$\begin{aligned} \underset{x_1,x_2}{\text{minimize}} \quad & f(x_1,x_2) = x_1 + 2x_2 \\ \text{subject to} \quad & h(x_1,x_2) = \tfrac{1}{4}{x_1}^2 + 5{x_2}^2 - 1 = 0 \end{aligned} \tag{5.15}$$

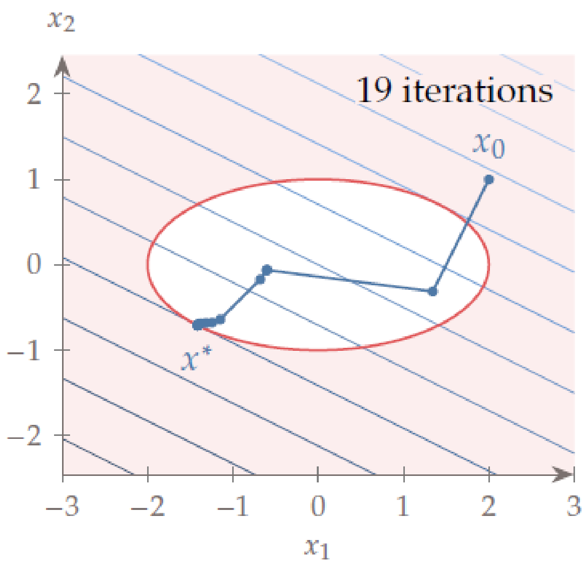


Figure 5.4: Path taken by the optimizer in a 2-variable problem (Equation 5.16) as given in [6] using inequality constraint.

Figure 5.4 shows the result that was obtained in the book [6], but instead of having it as an equality constraint, it is an inequality constraint. The red curve shows the boundary of the constraint and how the feasible points of optimality are within the curve. Figure 5.5 is the solution generated using the optimization algorithm described previously. It can be seen that here since the constraint has been taken as an equality constraint as described in Equation 5.15, it takes a path that lies outside the curve described by the constraint, and at the solution it converges onto the minimum that intersects with the curve, hence satisfying the equality constraint. But it can be seen that the optimal point found in this case is quite similar to the one obtained in [6].

Concerning the parameters in this case, again as stated previously, the parameter values would vary from problem to problem, and the only parameters that can be varied here are the ones associated with the line search method. In this case, $\rho = 0.5$ and $\mu_1 = 0.0001$ were used. However, if a merit function (Equation 5.6) is used during line search, an additional parameter can be varied, i.e. the penalty parameter μ_p. It can be recalled that as the value of this parameter increases the time taken for convergence would also increase as the path the optimizer would take would stick closer to the constraint curve.

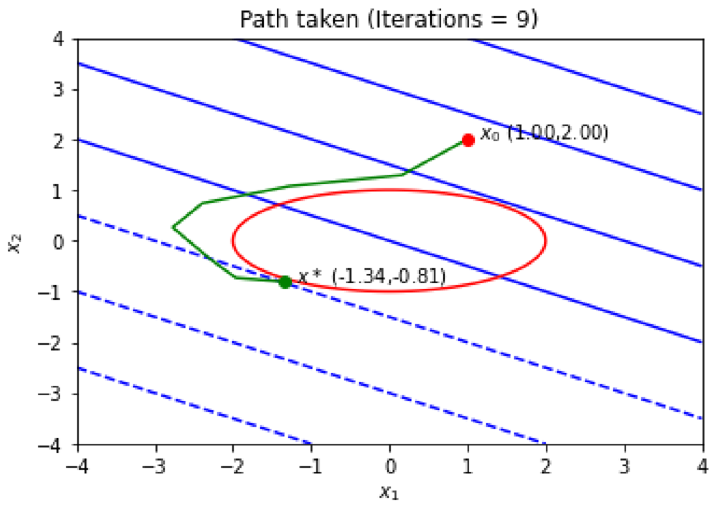


Figure 5.5: Path taken by the optimizer in a 2-variable problem (Equation 5.15).

5.3.3. Inequality constraint problem

For this test case again, the problem defined previously can be used, but instead of an equality constraint, it would now be an inequality constraint. The problem can be now written as.

$$\begin{aligned} \underset{x_1, x_2}{\text{minimize}} \quad & f(x_1, x_2) = x_1 + 2x_2 \\ \text{subject to} \quad & g(x_1, x_2) = \tfrac{1}{4}{x_1}^2 + 5{x_2}^2 - 1 \leq 0 \end{aligned} \tag{5.16}$$

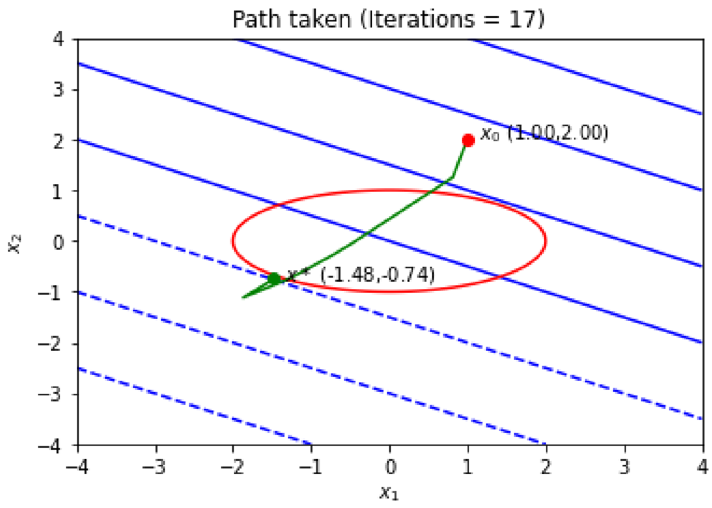


Figure 5.6: Path taken by the optimizer in a 2-variable problem (Equation 5.16).

As seen previously, Figure 5.4 shows the result obtained by Martins and Ning [6] and it was noted how the path taken by the optimizer is through the region within the constraint after starting from the outer region. The path converges onto an optimal point after 19 iterations. Now coming to the solution generated using the developed algorithm; Figure 5.6 shows the path followed by the optimizer after starting from a point outside the feasible region. The path converges onto a minimum after 17 iterations, and this was achieved by varying the parameters involved.

Looking at the parameters that can be varied, in this case, there are two additional parameters as compared to the Equality constraint case. Firstly it is the barrier parameter μ_b associated with the Logarithm of slack variables in Equation 5.2. As discussed previously, when the barrier parameter tends to zero the original objective is achieved. The effect of this parameter is that, when the initial value is high, the initial steps taken by the algorithm are quite large and can reduce the number of iterations if the corresponding optimal point also lies far away from the starting point. If the value selected is too large, the path taken would over-shoot in the initial directions and cause the number of iterations required for convergence to increase. Also If the value is too small, then the steps taken are also quite small again resulting in an increase in the number of iterations. Hence it may be important from case to case to find a correct balance for finding the initial value of the barrier parameter μ_b (these effects can be observed in Appendix B, Figure B.1). The value for the barrier parameter selected in this case is $\mu_b = 1$ Secondly, it can be seen that towards the end, the path does leave the feasible region and the reason behind this is that the penalty parameter (μ_p) associated with the Merit function (Equation 5.6) is not sufficient to keep the path within the constrained region. Again, if a very high value is selected for this, the number of iterations would also increase. But in this case, $\mu_p = 15$ was considered sufficient here because the main goal was to obtain a decrease in the number of iterations. Further, the line search parameters were kept the same as the previous case i.e., $\rho = 0.5$ and $\mu_1 = 0.0001$.

5.3.4. Equality and inequality constraint problem

The last test case is one where both, equality and inequality constraints will be used. A problem in the book of Martins and Ning [6] is defined where two inequality constraints are used, but here one of them will be turned into an equality constraint instead. The problem is given as,

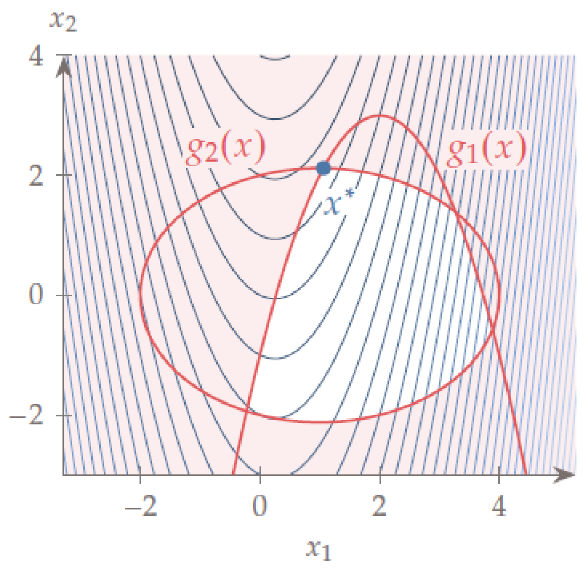


Figure 5.7: Optimization problem with two inequality constraints as given in [6].

$$\begin{aligned} \underset{x_1,x_2}{\text{minimize}} \quad & f(x_1,x_2) = x_1^2 - \frac{1}{2}x_1 - x_2 - 2 \\ \text{subject to} \quad & g(x_1,x_2) = x_1^2 - 4x_1 + x_2 + 1 \leq 0 \\ & h(x_1,x_2) = \frac{1}{2}x_1^2 + x_2^2 - x_1 - 4 = 0\,. \end{aligned} \tag{5.17}$$

Here in Figure 5.7, we can see that both the constraints given in Equation 5.17 are used as inequality constraints, but the optimal point, in this case, would be the same even if $g_2(x)$ as shown in the figure is modeled as an equality constraint.

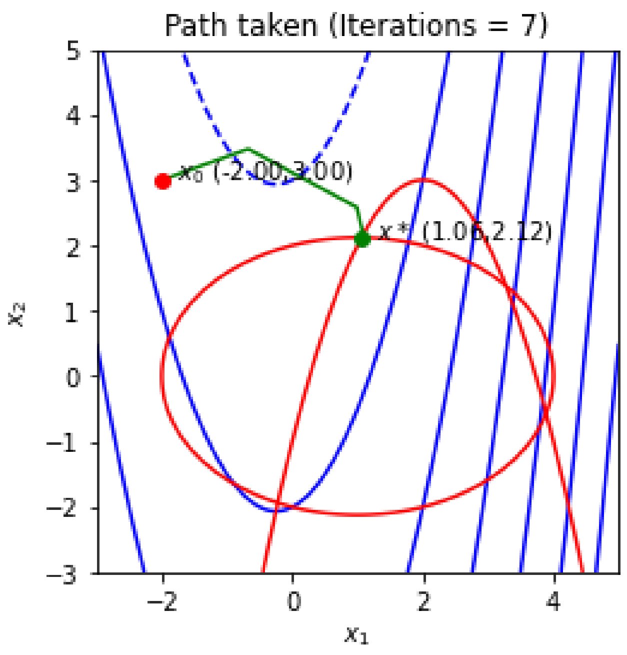


Figure 5.8: Path taken by the optimizer in a 2-variable problem with both, equality and inequality constraints (Equation 5.17).

After solving this problem using the developed optimizer, the result is shown in Figure 5.8. As would be expected since the elliptical curve is the curve formed by the equality constraint, the path that the optimizer takes is on the outside of the curve and moves its way to the optimal point. The parameters, in this case, are the same as the previous case of inequality constraint, and hence would entail the same explanation. In this case, the solution is achieved in 7 iterations.

5.4. Conclusion

In this chapter, a gradient-based optimization method is discussed and the algorithm used in this project is explained. The following conclusions can be drawn from what is covered so far,

1. Examples from the book of Martins and Ning [6] have been used as verification test cases. After having tested the optimization algorithm with all these cases, it can be concluded that the method works quite well and is also versatile.
2. The selection of the parameters involved can vary from one problem to another, and can be setup by observing the behavior of the optimizer in the first few iterations.

The next step would be the implementation of the deflation function to this method which will be discussed in the next chapter.

6 Deflation applied to optimization

The main scientific contribution of this thesis is presented in this chapter. The principle of deflation as discussed in chapter 3 will be extended to be applied to any optimization algorithm by carrying out a simple step. The goal is to enable the discovery of multiple local minima in non-convex optimization, as many as desired, which would eventually lead to the global minimum. A test case is presented in this chapter to illustrate how the different parameters affect the performance of the developed framework.

A Non-linear Programming (NLP) problem can be considered, for which multiple solutions need to be evaluated, and is given as [23],

$$\begin{aligned} \underset{x\in\mathcal{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c(x) = 0 \\ & l \leq x \leq u \end{aligned} \tag{6.1}$$

where l and u are the lower and upper bounds of the design variable x respectively. Now as seen in chapter 2, This constrained minimization problem can be converted into an unconstrained problem by expressing the objective function as the Lagrangian and is expressed as [23],

$$\mathcal{L}(x, \lambda, z_+, z_-) = f(x) + c(x)^T\lambda + (x-u)^T z_+ - (x-l)^T z_- \tag{6.2}$$

where $\lambda \in \mathcal{R}^m$ is the Lagrange multiplier vector associated with the equality constraints, $z_-, z_+ \in \mathcal{R}^n{}_+$ is the Lagrange multiplier vectors associated with the lower and upper bound respectively. Now, the KKT conditions are satisfied when [23],

$$\begin{gathered} \nabla_x \mathcal{L}(x, \lambda, z_+, z_-) = 0 \\ c(x) = 0 \\ l \leq x \leq u \\ z_+ \geq 0 \\ z_- \geq 0 \\ (x-u)^T z_+ = 0 \\ (x-l)^T z_- = 0 \end{gathered} \tag{6.3}$$

A point x that satisfies these conditions is therefore known as a KKT point, and finding multiple solutions to the NLP boils down to finding solutions that satisfy these conditions.

6.1. Deflation for optimization of NLP

The deflation operator and deflation matrix approach covered in chapter 3 can be applied to obtain multiple minima for the NLP described in Equation 6.1. The system of equations that satisfy the KKT conditions, herein called the KKT equations, can be expressed as a system of residuals given by $F(x, \lambda, z_+, z_-) = 0$ and systematically modified to obtain multiple minima. After finding the first solution x_1, the deflated system of KKT equations can be expressed as [23],

$$G(x, \lambda, z_+, z_-) \equiv M(x; x_1)F(x, \lambda, z_+, z_-) = 0 \tag{6.4}$$

Solving this problem with Newton-like methods requires the evaluation of the Jacobian of the deflated nonlinear system of equation G. To utilize the interior point algorithm introduced in chapter 5 the changes shown in Equation 6.4 needs to be implemented to the KKT equations of Equation 5.5. It must be noted that the Hessian of Lagrangian shown in Equation 5.5 is required to be a symmetric matrix to yield the corresponding descent direction. Tarek et al [23] found that after applying deflation, some numerical errors caused the Hessian matrix to be non-symmetric, resulting in numerical difficulties while calculating the search directions. This can possibly be remedied by the Hessian approximation shown in Equation 5.12 which forces the matrix to be symmetric.

But this method requires quite a few modifications that need to be carried out within the optimization method itself, and is probably a reason why Papadopoulos et al. [17] formulated a specialized primal-dual interior point algorithm from scratch to implement deflation. Doing such modifications to an entire method is quite tedious and the proposed adoption of deflation as an additional constraint, detailed in the next section, consists of a novel and simpler solution to the issue of the non-symmetric Hessian matrix.

6.2. Developing the deflation constraint

The idea to apply deflation as a constraint for optimization is original and independently developed in the context of this thesis. The only literature reference using a similar approach is the non-peer reviewed submitted pre-print [23]. In this pre-print the methodology consists of formulating an inequality deflation constraint that can be implemented in optimizers which support constrained optimization. However this methodology does not perform adequately in a larger design space after it has found the first few solutions (covered in test cases in section 6.3). For the present thesis, the approach is quite different because the deflation constraint is reformulated to include a crucial parameter that aids the constraint explore larger design spaces while yielding multiple distinct solutions. Hence, the method herein proposed can be considered as a novel method.

6.2.1. The initial approach

As stated previously, Tarek et al.[23] were the first to introduce the approach of implementing deflation as a constraint. In their work they propose two versions of the deflation constraint. In the first version, the work introduces an additional variable y into the optimization problem along with the deflation constraint m. With $y \geq 0$, the deflation constraint is given as [23],

$$m(x; x_1) = ||x - x_1||^{-p} + \sigma \leq y \tag{6.5}$$

where x_1 is the first optimal point that is found previously, p is a power (usually varies between 2-4), and a shift term σ (as introduced in chapter 3). After this constraint becomes active it is considered that, if the new found optimal point x_2 is associated with the variable y such that y is finite and has a value in exact arithmetic, then $x_1 \neq x_2$ (since $\lim_{\boldsymbol{x} \to \boldsymbol{x}_1} m(x; x_1) = \infty$). It can further be proved that every KKT point of the new NLP with a finite value of y is a KKT point of the original formulation [23].

A proof of how (x^*, y^*) would be a regular KKT point for the modified problem after the introduction of the deflation constraint is provided in Appendix C.

If it is assumed that $m > 0$ and $y > \sigma$, the deflation constraint would be equivalent to the distance constraint as follows [23]:

$$||x - x_1||^p \geq z \tag{6.6}$$

where $z = 1/(y - \sigma)$. Thus, it becomes clear that the deflation constraint essentially puts a constraint on the known solutions. If at any point, the optimizer approaches $z = 0$ or $y = \infty$, then the deflation operator is not distancing the new points from the known optimum points sufficiently. Hence, in this case, the main hyperparameters that can be varied to ensure that this distancing occurs are the shift σ and the power p as shown in Equation 6.5.

The method shown in Equation 6.5 can also be modified further to omit the additional variable y and replace it with a large finite constant M, such that the deflation constraint would look like [23],

$$m(x; x_1) = ||x - x_1||^{-p} + \sigma \leq M \tag{6.7}$$

Therefore, there are no additional variables, and with a proof similar to what is shown in Appendix C, it can be proved that the newly found x^* is a regular KKT point to Equation 6.1. So far, Equation 6.5 and Equation 6.7 have been evaluated with a single known solution, whereas for $\tilde{K}$ known solutions the deflation constraint can be expressed as a summation:

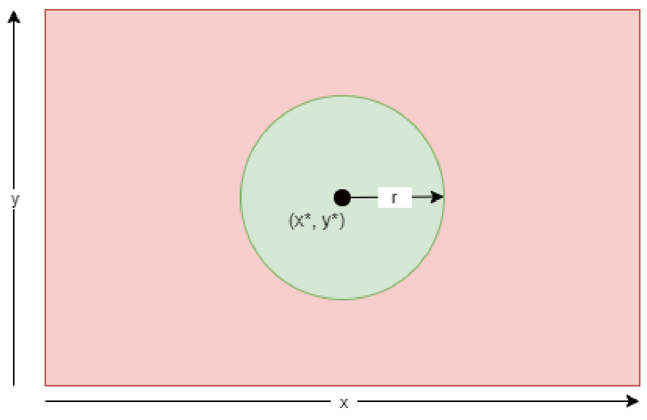
$$\sum_{k=1}^{\tilde{K}} m(x; x_k) \leq y \tag{6.8}$$

After implementing and testing these methods presented in [23], it was quite difficult to obtain the gradient-based optimizer to deflate away from the already found solution(s), and it did not seem to work as expected. There was a lot of dependence on the hyperparameters to be varied based on the number of required solutions. This warranted a further study of the parameters within this constraint, and how the constraint could be reformulated to have the desired results.

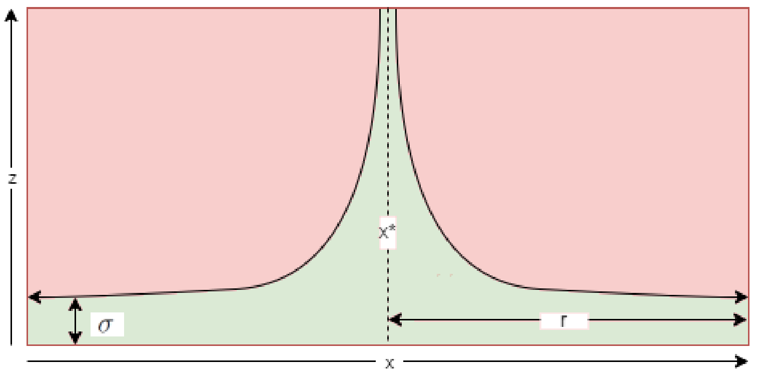
6.2.2. Improved deflation constraint

Based on the ideas for a deflation constraint as discussed previously, a new deflation constraint was developed and implemented in this thesis project. Firstly, it was important to clearly understand the parameters and how the constraint value varied with the parameters. In Figure 6.1, the variation of the value associated with the deflation constraint is shown about a known solution along the x and y axis i.e. (x^*, y^*). The curve it follows is a representation of the left-hand side of Equation 6.5 for $p = 2$. As the value of x approaches that of x^*, the value of the deflation function, $m(x; x^*) \rightarrow \infty$. The parameter σ that can be seen in Figure 6.1 is the value by which the curve of the deflation constraint gets offset or shifted. σ can be adjusted according to the height of the local maxima in the vicinity of the found solution to enable faster discovery of other local minima without getting stuck between these saddle points created by this constraint (This situation will be explained in section 6.3) while using a gradient-based optimizer. It must be noted that the value associated with the deflation constraint in such a formulation is always positive and in the previous section the deflation constraint was active or inactive mainly owing to the value of the variable y (Equation 6.8) through the optimization process.

Based on the observations of these parameters, another deflation constraint can be formulated such that it maintains a positive value within a given radius of action r and has been portrayed in Figure 6.1a. So



(a) Radius about which the constraint acts.



(b) Enlarged view of how the value of the deflation constraint changes away from the known solution.

Figure 6.1: A 2-D representation of the deflation constraint (Top and front plane).

a distance-based constraint can be formulated such that the constraint will be triggered by a conditional statement. For a known solution x_1, the constraint is given as,

$$\begin{array}{ll} \text{if} & ||x - x_1|| \leq r \\ \text{then} & m(x; x_1) = ||x - x_1||^{-p} + \sigma \end{array} \tag{6.9}$$

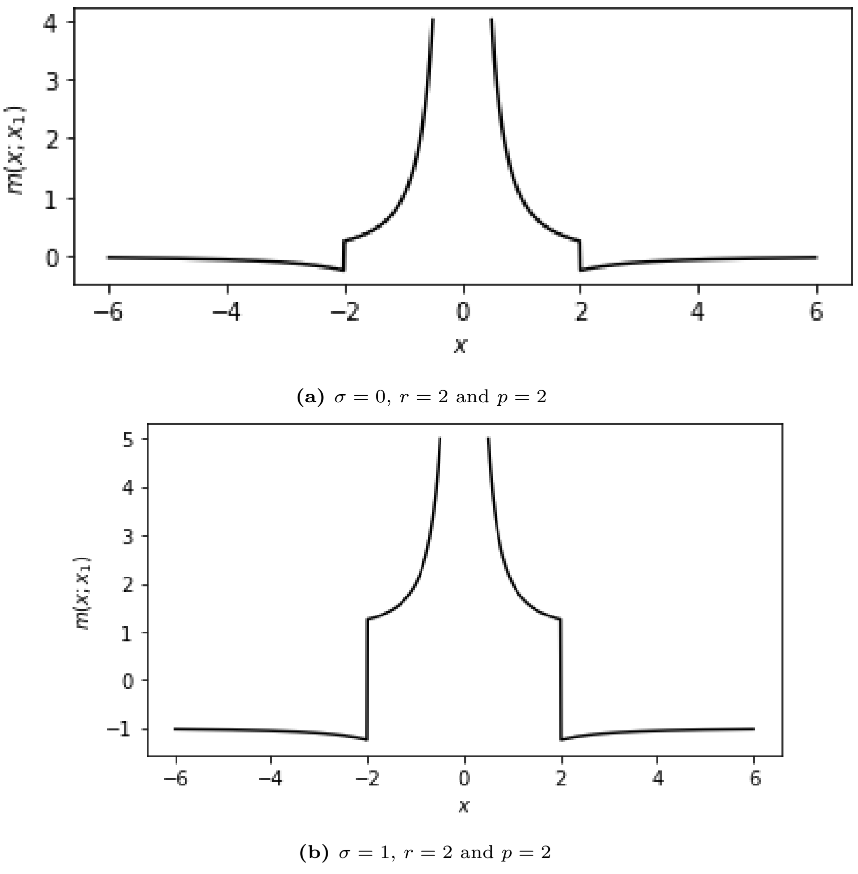
Since this constraint is triggered by a conditional, it implies that the constraint is only active when the optimizer is in the vicinity of the known solution. The advantage of this would be that, while forming the Lagrangian, if the conditional is not triggered, it reduces the need of allotting a Lagrange multiplier for the deflation constraint and additionally a slack variable as well in the case of an Interior point optimizer. Hence having a slight positive effect on computational efficiency.

The same constraint can also be formulated in a way that the need for a conditional statement can be omitted. The deflation constraint can be written as,

$$r_{dist} = r - ||x - x_1|| \tag{6.10}$$

$$m(x; x_1) = \frac{r_{dist}}{|r_{dist}|} * \left(||x - x_1||^{-p} + \sigma\right) \leq 0 \tag{6.11}$$

Hence this formulation acts as a traditional constraint and would require to have a Lagrange variable throughout the optimization process. Figure 6.2.2 provides plots of the variation in the deflation constraint values for two different σ values. For this plot, $r = 2$ and it can be seen that when r becomes 2, the constraint value crosses over to the negative and slowly tends to zero as the optimizer moves away from the known solution. Hence this constraint would be active only when the optimizer is within the radius of action r, i.e., when $m(x; x_1) > 0$. Hence it can be noted that the curves shown in Figure 6.1 and Figure 6.2.2 remain the same till the radius of action r is reached, beyond which the $\frac{r_{dist}}{|r_{dist}|}$ term in Equation 6.11 switches the value to negative.



(a) $\sigma = 0$, $r = 2$ and $p = 2$

(b) $\sigma = 1$, $r = 2$ and $p = 2$

Figure 6.2: The variation in value of deflation constraint represented in Equation 6.11 about a known solution $x_1 = 0$.

The plots for variation in parameters r, p have been attached in Appendix C. After having formulated the new deflation constraint, it was tested to verify its workings, which is covered in the section to follow.

6.3. Testing the deflation constraint in optimization

In this section, the discussed approaches to the deflation constraint will be tested, using a double-cosine function with multiple local minima, given in Equation 6.12 and plotted in Figure 6.3. The developed optimization scheme should be able to explore multiple or all of these local minima.

$$f(x_1, x_2) = -(cos(x_1)cos(x_2)) \tag{6.12}$$

Note that the double-cosine function is periodic, such that multiple maxima and minima span infinitely. Hence, while running the optimizer, bounds will be set for allocating a region of interest.

The optimizer used for running these tests is the gradient-based interior point optimizer discussed in chapter 5. The solution to Equation 6.12 using the different deflation constraint formulations is

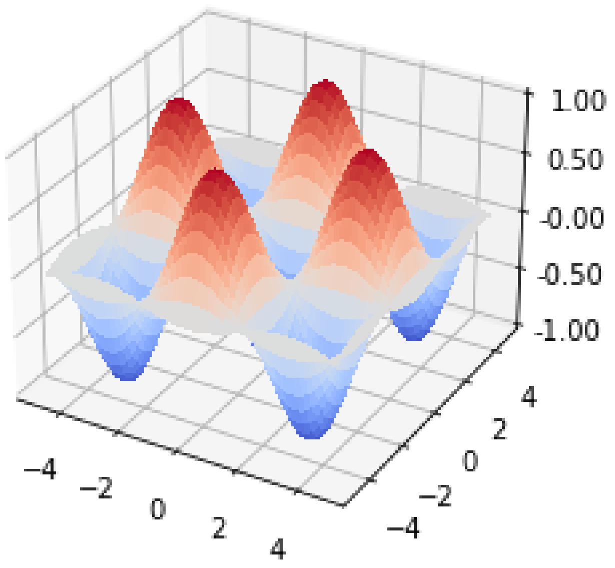


Figure 6.3: 3D plot of double-cosine function (Equation 6.12)

discussed below.

6.3.1. Approach using an additional variable

The first deflation constraint which was tested was the one presented in the work of Tarek et al. [23], as discussed in the previous section, and shown in Equation 6.5. This utilizes the additional variable y and is passed through the optimizer with the variables x_1 and x_2.

Point no.	x_1	x_2	y
1	-3.14159	-3.14159	1
2	-1.9626e-12	-1.9626e-12	11.3729
3	-0.167069	-0.167069	20.8866
4	-0.355569	-0.355569	20.799

Table 6.1: The results obtained while using additional variable y

The optimizer is set to minimize the double-cosine function within the bounds, (-4.8,4.8) along both axes and is provided with the starting point (-2.5,-2.5). The starting point is left to be the same while discovering the different solutions. The optimizer is also set to generate 4 solutions, which means that deflation has to be done thrice. As it can be seen in Table 6.1, the first point does not require the deflation constraint, hence the variable y has no value. The deflation constraint does work and moves to the second minimum at approximately $(0, 0)$. However, when it comes to the third minimum point, even though a finite value has been obtained for y, the optimizer is not getting deflated sufficiently away from the known solution, obtaining results that are very close to each other beyond this as can be seen in Figure 6.4 (where the black dots represent the obtained solutions and the red is the starting point), implying that the optimizer recognizes the region to be feasible. This would indicate that this formulation is causing the overall value of the deflation constraint $m(x, x^*)$ to be $\leq y$, keeping the deflation constraint inactive. The values of the parameters used in this case are stated under Figure 6.4, and varying these parameters did have an effect on deflation (portrayed in Appendix C). Increasing the σ value from 5 to 40, helped to deflate it further, and obtain the three minima along the diagonal with the fourth minima again in the vicinity of the third. Therefore it can be concluded that, if the value of y obtained through the optimizer is too large, it can cause the constraint to be inactive, hence

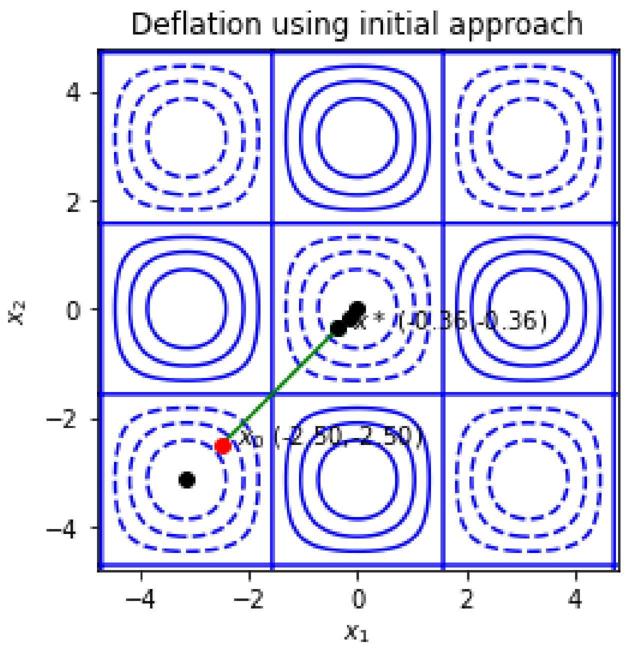


Figure 6.4: Deflation of double-cosine function (Equation 6.12) using additional variable (y) formulation as shown in Equation 6.5(deflated thrice with parameters, $\sigma = 5$, and $p = 2$).

preventing the deflation constraint from fulfilling its function.

6.3.2. Approach with a large finite value

This approach is another one discussed in the work of Tarek et al. [23] and is shown in Equation 6.7. Here, as mentioned previously, the additional variable y is replaced with a large finite value M. The starting point for the optimizer is the same as taken previously, i.e. (-2.5,-2.5), and is again set to generate 4 solutions. The M value was increased from the value 6 to observe how the solution changes (refer to images in Figure C.3). At $M = 10$ the deflation constraint progresses fairly well in locating three distinct minima, as can be seen in Table 6.2 and Figure 6.5, but at the 4th point is getting caught in the vicinity of the 3rd, hence the deflation constraint is becoming inactive around this point. The formulation of this constraint is such that an increase in the value of M would mean that the deflation constraint function m would reach inactivity sooner when compared to having a lower M value. This can also be observed in the images provided in Figure C.3. However, selecting the value of M would require a number of trials and seeing if it provides satisfactory results but it is important to note that exploring all the minima in this space would not be possible using this method and would require a sort of adaptive value assignment of M.

Point no.	$\mathbf{x_1}$	$\mathbf{x_2}$
1	-3.14159	-3.14159
2	1.23109e-07	1.23109e-07
3	3.14159	3.14159
4	3.32451	3.32451

Table 6.2: The results obtained while using finite value $M = 10$.

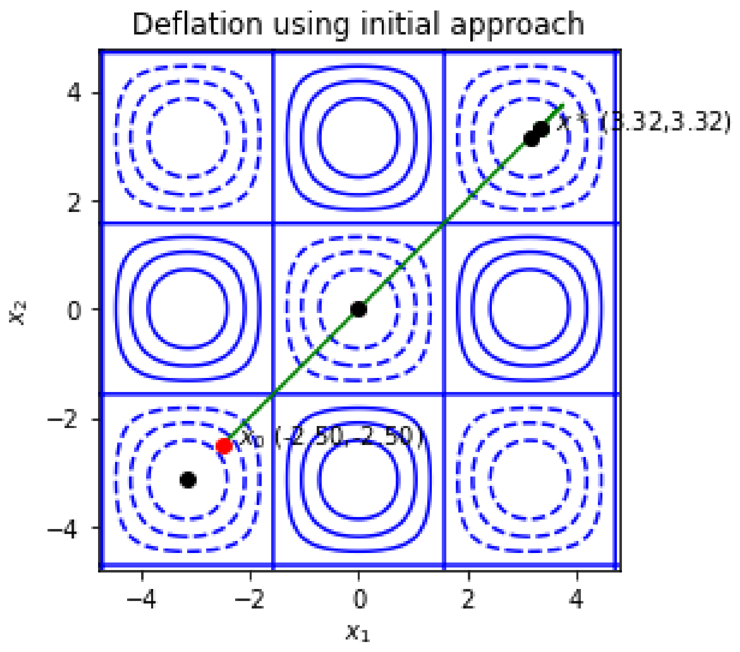


Figure 6.5: Deflation of double-cosine function (Equation 6.12) using finite value ($M = 10$) formulation as shown in Equation 6.7(deflated thrice with parameters, $\sigma = 5$, and $p = 2$).

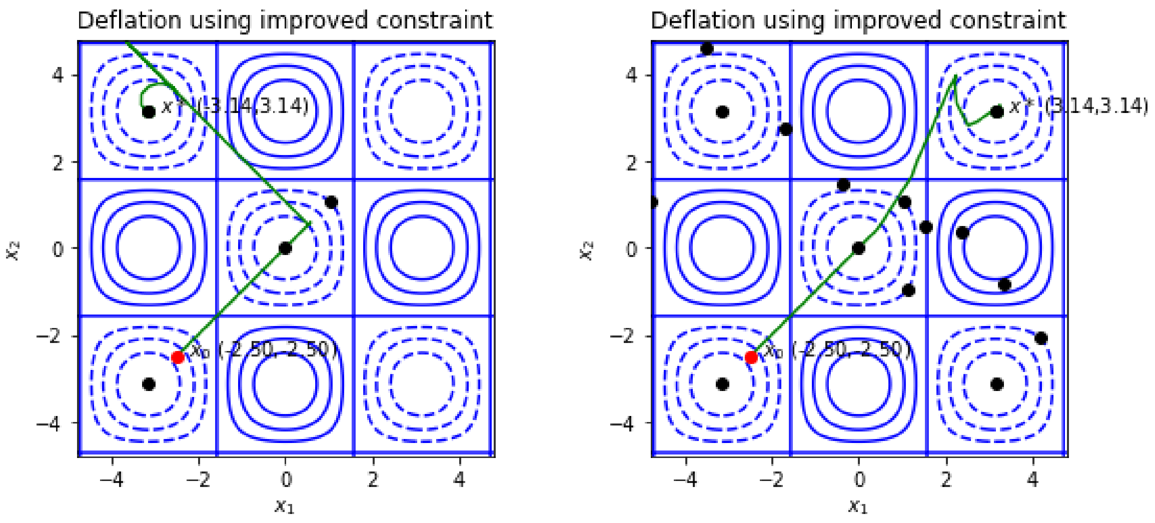
Point no.	x_1	x_2
1	-3.14157	-3.14157
2	-5.76301e-08	-5.76301e-08
3	1.06066	1.06066
4	-3.14159	3.14159

Table 6.3: The results obtained while using radius of action $r = 1.5$.

6.3.3. Improved deflation constraint

After understanding the shortcomings of the previously presented methods, the newly developed deflation constraint is tested with the double-cosine function. The constraint utilized is shown in Equation 6.9, and the same starting point $(-2.5, -2.5)$ is set for the optimizer. The additional parameter to be varied is the radius of action r, which for this case is chosen as $r = 1.5$. Therefore, it can be imagined that the deflation constraint remains active within a circle of 1.5 (units depending on the design variable) radius with the center being the known optimal point. Table 6.3 shows the optimizer was able to obtain 4 distinct minima. However, the third point is quite close to the second point and this can also be seen in Figure 6.6a. This does imply that the deflation constraint becomes inactive here, but there is a fairly intuitive reasoning behind this. The radius of action where the deflation constraint is active is 1.5, and the distance between these points would be equal to the radius of the action itself. The optimizer stops at this point because it can be imagined that where the deflation constraint is active is an upward-curved region with the maximum value of the deflation function being at the center of the circle i.e the known point. Around this region, the function value reduces, much like shown in Figure 6.2.2. And as the circle ends, the downward curve of the deflation function encounters the upward curve of the main objective function, hence creating a new local minimum at this point. This can be a shortcoming of this method, which could lead to the discovery of multiple of such minima, depending on the choice of the radius of action.

However, the main advantage of this method is that, even though not all the minima located would be of utmost significance, it can be used to locate them all, especially in this example of the double-cosine



(a) 4 solutions generated with $\sigma = 0$, $r = 1.5$ and $p = 2$ **(b)** 15 solutions generated with $\sigma = 0$, $r = 3$ and $p = 2$

Figure 6.6: The solution obtained after deflation of double-cosine function with the deflation constraint formulation given in Equation 6.9.

function. Figure 6.6b shows how all the significant minima of this objective function have been achieved after deflating 14 times, i.e. 15 distinct solutions obtained. Increasing r, in this case, proves to improve the number of evaluations required to find the significant minima, but after a certain threshold, if the radius becomes too large, it may engulf significant minima, hence deflating away from this region (this is covered in Appendix C under Figure C.3). Therefore, utilizing a smaller radius of action might lead to more evaluations being required, but in the end, it makes the algorithm more robust by enabling the discovery of all possible minima.

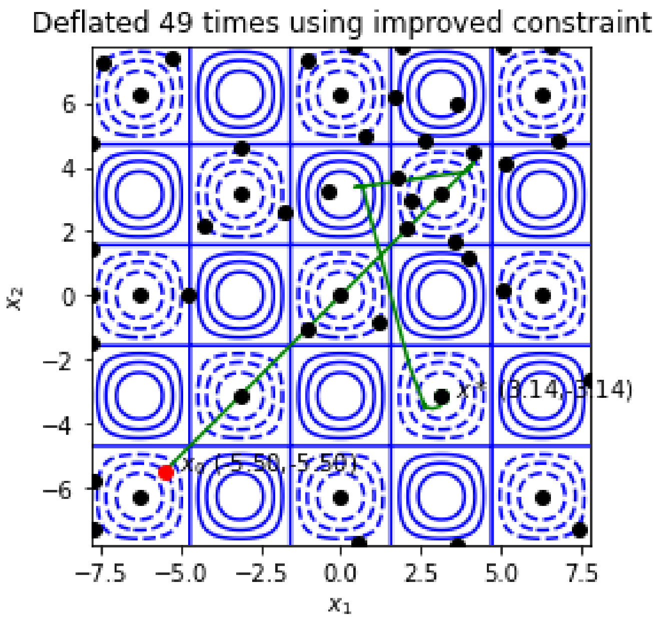
6.3.4. Updating the optimizer's starting point

The concept of deflation was developed while considering that the starting point of the optimizer remains the same and hence even starting at the same point leads to the discovery of different solutions. However, while testing the deflation method described previously, it was hypothesized that once the gradient-based optimizer has found a minimum in one region around its starting point, having a new starting point from this minimum for the next round of optimization would enable the next minimum to be found with more computational efficiency.

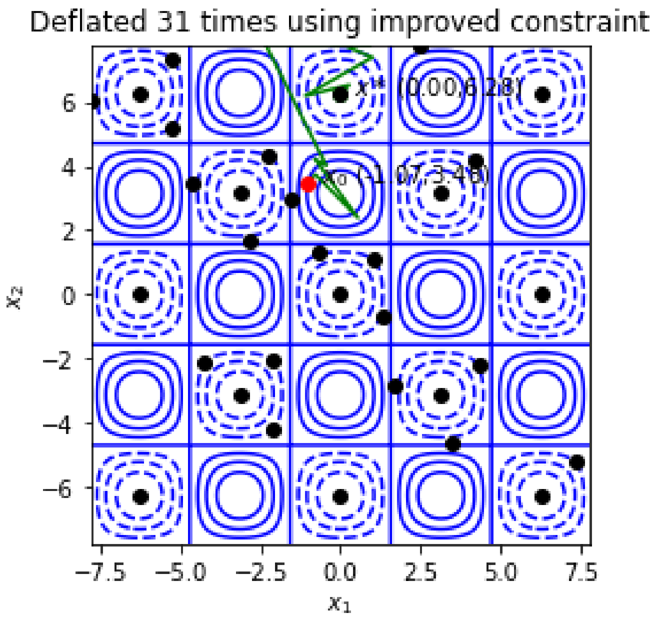
This hypothesis was tested using the same double-cosine function (Equation 6.12, but this time the bounds of the optimization space were enlarged such that both axes spanned from $(-7.8 \rightarrow 7.8)$ and the initial starting point was set as $(-5.5, -5.5)$. The parameters of the deflation constraint are set as; the radius of action $r = 1.5$, shift $\sigma = 1$, and power $p = 2$.

Firstly a baseline was set using the case where the starting point was kept the same as seen in Figure 6.7a. It can be seen that to find all the significant minima 50 solutions need to be evaluated, and in these 50 solutions, the optimizer comes across many other local minima which are either created by the deflation constraints associated with the already found solutions as explained in the previous section, or saddle points of the objective function.

For the second case, as shown in Figure 6.7b the only change made was for the start point of the optimizer after finding a new solution. The new start points were set to be points with a small offset from the point of the previous solution, because at the exact point of the previous solution the deflation constraint value becomes $m(x, x_k) = \infty$, which would raise numerical difficulties. To illustrate this idea, if the first minimum is given by the coordinates (2, 2), the starting point after the first deflation can



(a) 50 solutions generated with same starting points ($\sigma = 1$, $r = 1.5$, $p = 2$ and time = 47.84 sec)



(b) 32 solutions generated with updated starting points ($\sigma = 1$, $r = 1.5$, $p = 2$ and time = 14.45 sec)

Figure 6.7: The effect of updating the starting point after finding each new minimum and performing a new deflation.

be (1.9, 2) or (2.1, 2). In Figure 6.7b, it can be observed that the optimizer can locate the significant minima in just 32 evaluations as compared to the previous 50 evaluations when starting from the same point, hence resulting in way fewer discoveries of the undesirable minima. There is no established metric as of now with how the number of evaluations would be affected by a change in the size associated with the design space, but it could be an interesting study to help improve the efficiency of this method.

6.4. Conclusion

In this chapter, the concept of deflation applied as a constraint was explored, first by reviewing the only work present on this concept done by Tarek et al. [23]. This concept was further built on, and an improved way of implementing a deflation constraint has been developed in this project. The following conclusions can be drawn from what has been discussed so far,

1. Applying deflation as a constraint opens up opportunities to explore different solutions to a non-convex problem, while being compatible with any optimization algorithm that supports the addition of a new constraint.
2. With the formulation of an inequality deflation constraint to find multiple minima, an originally unconstrained problem would become a constrained problem when the deflation constraint is added.
3. Applying the deflation constraint to gradient-based optimizers helps to further explore the capabilities of these optimizers to find global minima as compared to gradient-free ones, such as genetic algorithms or particle swarm, where accuracy could be lost due to their heuristic nature.
4. The methods proposed in the pre-print of Tarek et al. [23] do provide a foundation for the idea of using deflation as constraint. However, in the attempts carried out using these methods, the obtained results were not satisfactory. The results were sensitive to the parameters affecting the methods, for instance y in Equation 6.5 and M in Equation 6.7. However, even after varying these parameters, the constraint would fail to deflate the optimizer beyond a certain minimum, hence limiting the number of meaningful solutions that could be obtained.
5. The improved deflation constraint proposed in this thesis proved to perform well and robustly, fulfilling its function to find all minima points in a design space. The formulation is also dependent on its main parameter, the radius of action r. If the value of r is too large, it could lead to the constraint engulfing significant minima points. Smaller values for r are recommended, even though more function evaluations may be needed. Hence, to further improve on the method, a study could be done to make the value of r to change dynamically, or to revisit the current constraint formulation while trying to remove the dependence on r.

This page was intentionally left blank.

Part III

Results

7 Case studies

In this chapter, the methodology discussed in the previous chapters is used to assess the working of the optimization method along with the deflation constraint while comparing it to pre-existing optimization methods. Additionally, the deflation constraint will also be used in one of these pre-existing optimization methods to demonstrate its capability in gradient-free optimization as well. The following case studies related to composite design will be covered in this chapter:

1. **Case study 1:** The optimization of laminate stacking sequence for buckling load maximization is based on the work of Riche et al. [61].
2. **Case study 2:** The optimization of laminate stacking sequence for a minimum weight and a required buckling strength using deflation coupled with the pre-existing optimization method.
3. **Case study 3:** The optimization of lamination parameters for a given cantilever loading on a variable stiffness composite plate based on the work of Setoodeh et al. [62].
4. **Case study 4:** Retrieval of fiber paths from optimized lamination parameters which will be compared to the work of Setoodeh et al in [50].
5. **Case study 5:** Multi-layer variable stiffness plate fiber path retrieval using the developed method in comparison to the pre-existing method of optimization.

The case studies will be explained in sub-sections, where the results obtained using the developed deflated gradient-based optimization algorithm will be compared to a pre-existing genetic algorithm optimizer that is available in the Python module, pymoo [63].

7.1. Case study 1

7.1.1. Problem description and formulation

This case study based on the work of Riche et al. [61] serves as an example of how the deflation method can provide quite versatile solutions for a certain problem and can be utilized to explore other options if the global minimum is undesirable from a manufacturing point of view. The fiber angles for a constant stiffness and constant mass plate under bi-axial loading will be optimized, with the objective of maximizing the buckling load. The geometry of the plate under consideration is shown below in Figure 7.1. The figure depicts a simply supported plate with bi-axial loading; λN_x along the x-axis and λN_y along the y-axis with λ being an amplitude parameter. The length and width of the plate are respectively a and b. The laminated plate is considered to have N number of plies, and is assumed to be symmetric and balanced, using ply angles of $0°, 90°$, and $\pm 45°$. The thickness (t) of each ply is assumed to be the same. The symmetric and balanced condition provides a reduction in the number of design variables, hence each ply does not have to be attributed to a design variable. Thus, for a laminate with N plies, the number of independent design variables is $N/4$.

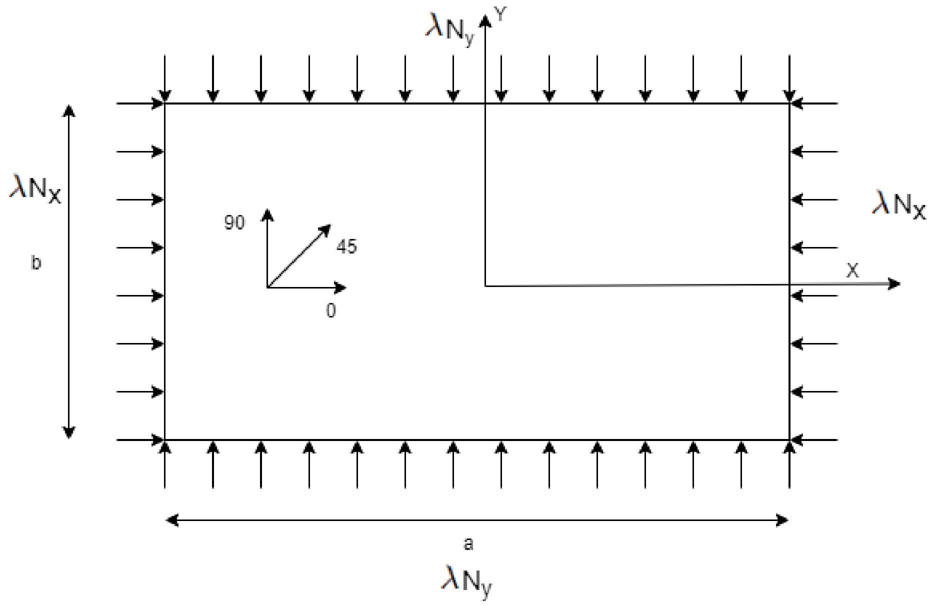


Figure 7.1: The geometry of laminated plate and applied loads [61].

For calculating the critical buckling eigenvalueλ_{cb}, it is assumed that the laminate buckles with m half waves in the $x-$direction and n half-waves in the $y-$direction. Hence, an analytical solution can be derived such that the buckling load amplitude λ_b is a function of the out-of-plane stiffness matrix of the laminate (D_{ij}) and the number of half waves in both, x and y directions. Riche et al. have provided a relation for this in their work [61]. However, due to a possible misprint, the relation is incorrect. This relation was then corrected by deriving the relation again with reference to the book of Kassapoglou [1] and the correct relation is given as,

$$\frac{\lambda_b(m,n)}{\pi^2} = \frac{[D_{11}(m/a)^4 + 2(D_{12} + 2D_{66})(mn/ab)^2 + D_{22}(n/b)^4]}{(m/a)^2 N_x + (n/b)^2 N_y} \tag{7.1}$$

The values of m and n need to be found such that it minimizes λ_b to yield the critical buckling load λ_{cb} and this varies with the number of plies considered, the material, the dimensions of the plate and the load case. Now, for the constraints; in this optimization case, the allowable strains in each layer were set as a constraint, hence limiting the strain in each layer. The ultimate allowable strains are $\epsilon_1{}^{ua} = 0.008$, $\epsilon_2{}^{ua} = 0.029$, and $\gamma_{12}{}^{ua} = 0.015$. The principle strains in the ith layer of the laminate are related to the applied loads as follows [61],

$$\begin{aligned} \lambda N_x &= A_{11}\epsilon_x + A_{12}\epsilon_y \\ \lambda N_y &= A_{12}\epsilon_x + A_{22}\epsilon_y \end{aligned} \tag{7.2}$$

$$\begin{aligned} \epsilon_1^i &= \cos^2\theta_i\epsilon_x + \sin^2\theta_i\epsilon_y \\ \epsilon_2^i &= \sin^2\theta_i\epsilon_x + \cos^2\theta_i\epsilon_y \\ \gamma_{12}^i &= \sin^2\theta_i(\epsilon_y - \epsilon_x) \end{aligned} \tag{7.3}$$

In Equation 7.3, θ_i refers to the fiber angle associated with the ith layer with ϵ_x and ϵ_y are the global strains of the plate. The term λ in Equation 7.2 is the load amplitude and this becomes λ_{cs} for the lowest value of λ such that only one of the principal strains in one of the layers has exceeded the allowable strain. In the case herein considered, γ_{xy} is not present because the layers of the laminate are to be balanced and symmetric.

The optimization is done to maximize the buckling load, and in this case there are no ply contiguity constraints taken into account. Hence, the optimization problem can be formulated as,

$$\begin{aligned} \underset{\theta \in \mathcal{R}^n}{\text{minimize}} \quad & (1/\lambda_{cb}) \\ \text{subject to} \quad & {\epsilon_1}^i \leq 0.008 \\ & {\epsilon_2}^i \leq 0.029 \\ & {\gamma_{12}}^i \leq 0.015 \end{aligned} \tag{7.4}$$

The objective function is given as shown in Equation 7.4 because it is a maximization problem that can be formulated as an inverse of a minimization problem. The results are generated for a graphite-epoxy plate with the following material properties and ply thickness; $E_1 = 18.50E6$ psi (127.59 GPa), $E_2 = 1.89E6$ psi (13.03 GPa), $G_{12} = 0.93E6$ psi (6.41 GPa), $\nu_{12} = 0.3$, and $t = 0.005$ in. (0.0127 cm).

	Load factor λ	
Stacking sequence	**Buckling (cb)**	**Failure (cs)**
$(90_{10}, \pm45_2, 90_2, \pm45_3, 90_2, \pm45_4)_s$	3973.01	14,205.18
$(\pm45, 90_{10}, \pm45, 90_8, \pm45, 90_8)_s$	3973.01	8,935.74
$(90_4, \pm45_2, 90_{16}, \pm45, 90_6)_s$	3973.01	8,935.74
$(90_2, \pm45, 90_6, \pm45, 90_8, \pm45, 90_{10})_s$	3973.01	8,935.74
$(90_8, \pm45, 90_2, \pm45, 90_2, \pm45, 90_2, \pm45_6)_s$	3973.01	14,205.18

Table 7.1: Results for the buckling load maximization problem obtained by Riche et al. [61].(64 plies, $a = 20$ in., $b = 10$ in., $N_x = 1$ lb, $N_y = 1$ lb)

7.1.2. Results

The Table 7.1 shows the results that were obtained by Riche et al., and it can be observed that they have generated multiple half stacks with the same buckling load and different strain failure loads. This portrays the variety of combinations that can be obtained for the same design load and in this way by assessing the options a feasible stacking sequence can be selected. It can be observed that the load factors for strain failure are consistently higher than the critical buckling load factors which imply that the strain constraint in each layer is not violated. These results were obtained using Genetic Algorithm (GA) that allows for parallel processing. It must be noted that the GA had to make combinations considering only the three allowed ply angles, i.e. $0°, 90°$, and $\pm45°$. Whereas, for the gradient-based optimizer, the fiber angle is varied continuously between (0, 90). After the optimized stacking sequence is found, the results are manipulated such that, the interval (0,90) is divided into 3 parts which are, (0, 30), (30, 60), and (60, 90). If the ply angle corresponds to the first second or third interval, the value is set to $0°, 90°$, and $\pm45°$ respectively.

The gradient-based optimizer with the deflation constraint provided a variety of stacking sequences with similar critical buckling load factors, as compared to the results obtained by Riche et al., after generating 80 solutions (which took 23 minutes). Table 7.2 shows the 5 best solutions that were obtained, where it can be seen that the results may be close to those obtained by Riche et al. but are still inferior. Hence the best result cannot be attributed as possible as a global optimum. The reason behind this could be because the gradient-based optimizer considers continuous variables as mentioned previously, opposed to the three discrete fiber angles utilized by GA. Therefore, it can be considered as another drawback to the gradient-based deflation method, that discrete variables cannot be taken into account

	Load factor λ	
Stacking sequence	**Buckling**	**Failure**
$(90_6, \pm45, 90_4, \pm45, 90_2, \pm45, 90_2, \pm45, 90_2, \pm45, 90_4, \pm45, 0_2, 90_2)_s$	3955.44	8,072.14
$(90_8, \pm45_2, 90_2, \pm45, 90_2, \pm45, 90_4, \pm45, 90_4, \pm45)_s$	3954.9	11,619.78
$(90_4, \pm45_2, 90_{24})_s$	3953.88	8,027.76
$(90_12, \pm45_20)_s$	3945.61	15,029.39
$(90_6, \pm45, 90_4, \pm45, 90_4, \pm45_2, 90_10)_s$	3923.9	9,837.9

Table 7.2: Results for the buckling load maximization problem obtained using the gradient-based optimizer with deflation. (Top 5 of 80 solutions generated with $p = 2$, $\sigma = 1$ and $r = 8$)

for optimization and conversion of optimized variables to required discrete values can yield points that are not actual local minima.

7.2. Case study 2

7.2.1. Problem description and formulation

In this case study, minimization of weight is explored with reference to the work of Riche et al. [61]. Here the thickness of the plate being optimized also becomes a design variable and the objective function is the volume of the plate. To attribute to the thickness of the plate, another set of design variables is included in addition to the ply fiber angles, i.e. the thicknesses associated with each ply. This doubles the number of design variables. Schläpfer [64] introduces the concept of ghost layers, which is essentially a layer that carries information about material properties and also the ply fiber orientation but is associated with zero thickness value. Hence if the jth layer is zero, the following must be applicable for a plate with j layers,

$$\sum_{k=1}^{j} t_k - \sum_{k=1}^{j-1} t_k = 0 \text{ for } t_j = 0 \tag{7.5}$$

Now for this case, the objective function is the volume of the plate and additionally compared to the previous case, another constraint can be introduced into the problem which is the buckling load factor of the plate. The allowable strain constraint will also be appended such that it can satisfy a required target load factor. This target load factor would be set even for the buckling load factor. Hence the optimization problem can be formulated as follows,

$$\begin{aligned} \underset{\theta, t \in \mathcal{R}^n}{\text{minimize}} \quad & (\sum_{i=1}^{j} t_i)ab \\ \text{subject to} \quad & {\epsilon_1}^i \leq 0.008 \\ & {\epsilon_2}^i \leq 0.029 \\ & {\gamma_{12}}^i \leq 0.015 \\ & 1 - (\lambda_{cb}/\lambda_t) \leq 0 \\ & 1 - (\lambda_{cs}/\lambda_t) \leq 0 \end{aligned} \tag{7.6}$$

Here t_k is the thickness associated with each ply, a is the length of the plate, b is the width (refer, Figure 7.1) and the new term in the constraints, λ_t is the target load factor that needs to be achieved with the primary objective being volume optimization. The material properties are the same as utilized in the previous case study, for a graphite epoxy plate. The baseline can be established by referring to another set of results by Riche et al. [61] where they optimize a plate with 48 with the objective being buckling load factor maximization. Table 7.3 shows 4 stacking sequences with 48 plies with the corresponding load factors. So for a target load factor that is lower than these optimized values of load factors, it can be hypothesized that the number of plies required would also reduce, hence lowering the volume.

	Load factor	
Stacking sequence	**Buckling**	**Failure**
$(90_2, \pm45_4, 0_4, \pm45, 0_4, \pm45, 0_2)_s$	14,168.12	13,518.66
$(\pm45_3, 0_2, \pm45_2, 0_2, 90_2, 0_4\pm45, 0_2)_s$	14,134.76	13,518.66
$(90_2, \pm45_3, 0_2, \pm45, 0_2, \pm45, 0_4, \pm45, 0_2)_s$	14,013.71	13,518.66
$(\pm45_2, 0_2, \pm45_2, 90_2, 0_4, \pm45, 0_2, \pm45, 0_2)_s$	13,662.61	13,518.66

Table 7.3: Results for the buckling load maximization problem obtained by Riche et al. [61]. (48 plies, $a = 20$ in • $b = 5$ in, $N_x = 1$ lb, $N_y = 0.125$ lb).

7.2.2. Results

Since it was concluded from the previous case study that, the gradient-based optimizer is not the best to be used with discrete-valued variables, in this case, a pre-existing Genetic Algorithm optimizer that is available in the python module, pymoo [63] was utilized. For this optimization method, the deflation constraint is also included to obtain multiple solutions and to see its working with a different optimizer to test the claim that it can be run with any optimization method. For this optimizer, once again the variables need to be continuous, but since it is a heuristic method, the conversion of continuous to discrete variables can be done during the optimization process itself and this would directly affect the value of the objective function, unlike the gradient-based method. So this in turn makes it possible for the method to be applied for discrete variable optimization.

The main parameters to be varied in this optimizer are the population size, that is the number of individuals that is crosses over to get the fittest individual, and the number of generations, which is used as a termination criterion. In this case, it is known what the ballpark figures for the load factors associated with a stacking sequence with 48 plies look like (Table 7.3), so the target λ_t is set as 10000, which should result in a decrease in the number of layers. So the optimizer is given a total number of variables as 24 which is 12 for the ply angles (note balanced and symmetric) and 12 for the thicknesses associated with the 12 layers. While generating the results, a new term is introduced, that is the margin of safety, which is a margin over the target load factor and can be expressed as a percentage. This is calculated as,

$$\begin{aligned} MS_{cb} &= (\lambda_{cb}/\lambda_t) - 1 \\ MS_{cs} &= \lambda N_y = (\lambda_{cs}/\lambda_t) - 1 \end{aligned} \tag{7.7}$$

		Margin of Safety	
Stacking sequence	**Plies**	**Buckling**	**Failure**
$(90_4, 0_6, 90_2, \pm45_5)_s$	48	7.44%	15.85%
$(\pm45_3, 0_4, \pm45_2, 0_2, \pm45_2, 0_2)_s$	44	6.49%	12.94%
$(\pm45_4, 0_4, \pm45, 0_2, \pm45, 0_2, \pm45)_s$	44	10.3%	12.94%
$(90_2, 0_6, 90_4, 0_2, 90_4, \pm45_3)_s$	48	3.84%	9.21%
$(90_4, 0_4\pm45_2, 90_2, 0_2, 90_4, 0_2, 90_2)_s$	48	13.73 %	5.155 %
$(90_2, \pm45, 0_2, \pm45, 0_2, \pm45_4, 0_4)_s$	44	2.55%	14.2%

Table 7.4: Results for the buckling load maximization problem obtained using the GA optimizer with deflation. (6 solutions generated with $p = 3$, $\sigma = 1$ and $r = 1$)

The first run with the genetic algorithm was done without deflation, and with the number of generations starting at 10. Despite varying the number of generations to 50, there were no changes in the number of plies and gives the first solution in Table 7.4. Then the population size was changed from 20 to 30 and a 44-plies layup was obtained, with 8.7% and 13.08 % margin of safety for buckling and strain failure load factors respectively. Following this, the number of generations was turned down to 30, with the populations size back to 20, and now the deflation constraint was applied to obtain 6 solutions. After deflation, it can be seen that a variety of solutions have been obtained with a varying margin of safety and also a few with decreased number of plies. The third solution seems to be the most promising with the maximum margin of safety obtained for the critical buckling load factor and can seem better than the one obtained with GA alone after varying the parameters.

Hence it can be concluded that applying deflation to this GA optimization proves that the deflation constraint works even in heuristic algorithms for optimization and it helps to explore the design space better to obtain a global optimum even if the parameters of the algorithm are not at their optimal values.

7.3. Case study 3

7.3.1. Problem description and formulation

The development of variable stiffness laminates was the primary motivation behind this thesis project in the course of which an optimization algorithm was developed. Variable stiffness can be modeled in a structure either by using Lamination parameters or by using a direct angle parameterization approach as we have seen in the chapter 4. After seeing the potential in Lamination parameters, it was decided to explore if the developed method can be used to optimize lamination parameters. The minimum compliance design of VS panels done by Setoodeh et al. [62] is going to be verified in this case study. This can be achieved through the minimization of the complementary strain energy. The in-plane or bending behavior of a composite plate can be modeled using only 4 parameters regardless of how many layers there are and whether the feasible LP domain is convex. It can be proved that the complementary strain energy in the LP space is also convex [62]. Hence the local design problem would also be convex. In this case, the distribution of the 4 in-plane LPs will be obtained for a cantilever plate that is under a uniformly distributed load q_0.

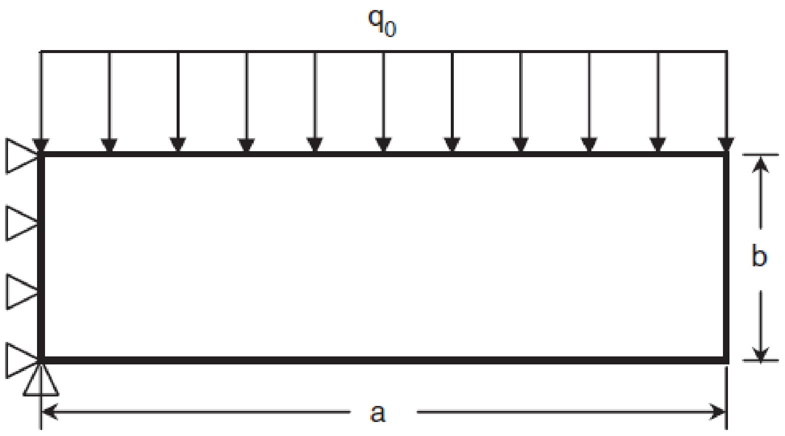


Figure 7.2: Cantilever plate with uniform load [62].

The problem is solved by discretizing the plate into multiple elements and using Finite Element Analysis to get the displacements at each of the element's nodes. To do this the python module pyfe3d [65], developed by Castro was implemented which also has great support for composites in this module. The displacements were used to then calculate the strain components at each node. An important aspect of getting the strains from the displacements obtained from the FEM is multiplying the displacement by strain-displacement matrix B_L (if interested, the reader can refer to Chapter 6 in the book by Logan [66] for its derivation). After finding the strains, the nodal strains are multiplied by the in-plane and coupling matrix (A-B matrix which can be calculated as shown in Equation 4.13) corresponding to its nodal values, to get the nodal stress resultants, N_x, N_y, and N_{xy}. Now the minimization problem is formulated as,

$$\begin{aligned} \underset{V_i}{\text{minimize}} \quad & \tfrac{1}{2} N_i^T \cdot A^{-1}(V_i) \cdot N_i \\ \text{subject to} \quad & 2V_1^2\,(1 - V_3) + 2V_2^2\,(1 + V_3) + V_3^2 + V_4^2 - 4V_1V_2V_4 \leqslant 1 \\ & V_1^2 + V_2^2 \leqslant 1 \\ & -1 \leqslant V_3 \leqslant 1 \end{aligned} \tag{7.8}$$

where N_i is the vector of the resultant forces for the Node i, A is the in-plane stiffness matrix which is a function of V_i calculated with Equation 4.13 and $V_i = \{V_{i_1}, V_{i_2}, V_{i_3}, V_{i_4}\}$, the vector of in-plane LPs of the ith node. For balanced VS laminates, the LPs $V_2 = V_4 = 0$. The material properties used for the numerical evaluations are, $E_{11} = 181.0\,GPa$, $E_{22} = 10.3\,GPa$, $G_{12} = 7.17\,GPa$, and $\nu_{12} = 0.28$.

7.3.2. Results

The results in this optimization are obtained for each node with 4 Lamination parameters. Hence it would be feasible to view these distributions across the nodes as contour plots and to make a comparison of how well the objective function of the entire plate, the compliance is minimized. A non-dimensional compliance term can be compared which is given as [62],

$$\bar{C} = \frac{E_{22} h b^3 C}{q_0^2 a^5} \tag{7.9}$$

where E_2 is the transverse modulus of elasticity, h is the thickness of the laminate, b is the width of the laminate, q_0 is the uniformly distributed load, a is the length of the laminate and C is the compliance obtained after Optimization and is given as,

$$C = \frac{1}{2} F^T \cdot U \tag{7.10}$$

where F and U are the vectors of external forces and displacements respectively.

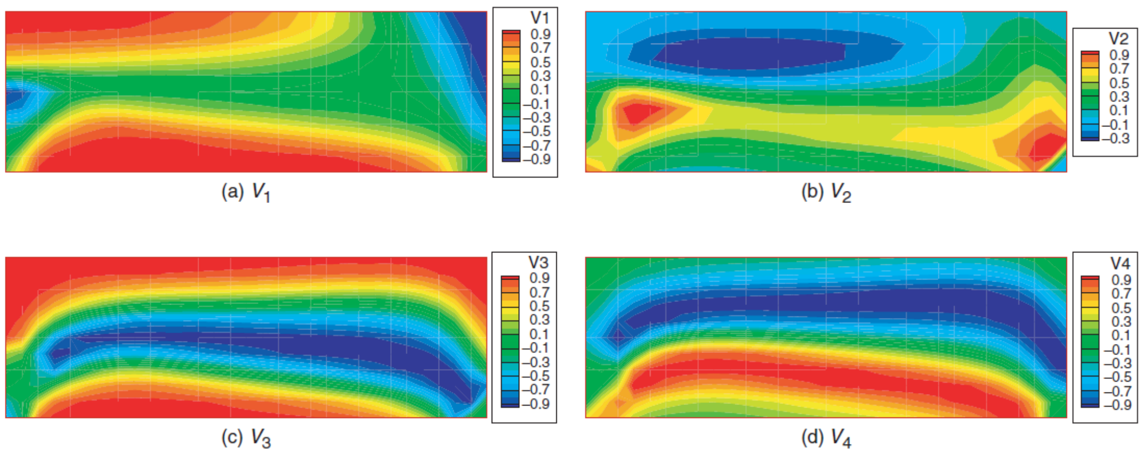
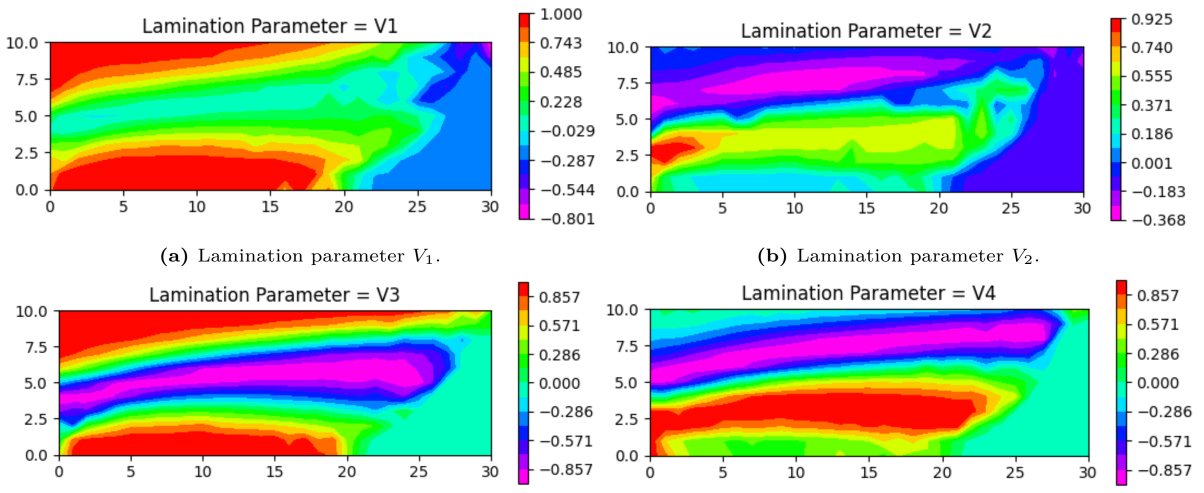


Figure 7.3: Optimal Lamination parameter distribution for Cantilever plate [62]. ($a/b = 3$, and 31×11 nodes)

The results obtained by Setoodeh et al. [62] will be the baseline for comparison using two different optimizers; The gradient-based optimizer to see if the convex design space can be used to an advantage, as well as the GA from the pymoo python module as seen previously. Figure 7.3 shows the distribution of the 4 optimal LPs across the laminate's nodes and can be visually compared to the results obtained using the gradient-based interior point optimizer shown in Figure 7.4. Visually, it seems like the result generated by the gradient-based optimizer is similar to the ones obtained by Setoodeh et al., but has some dissimilarities towards the right bottom corner. This could be attributed to the fact that at that portion the stress acting is minimal since it is a free end. Table 7.5 shows the non-dimensional compliance values calculated using Equation 7.9.



(a) Lamination parameter V_1.

(b) Lamination parameter V_2.

(c) Lamination parameter V_3.

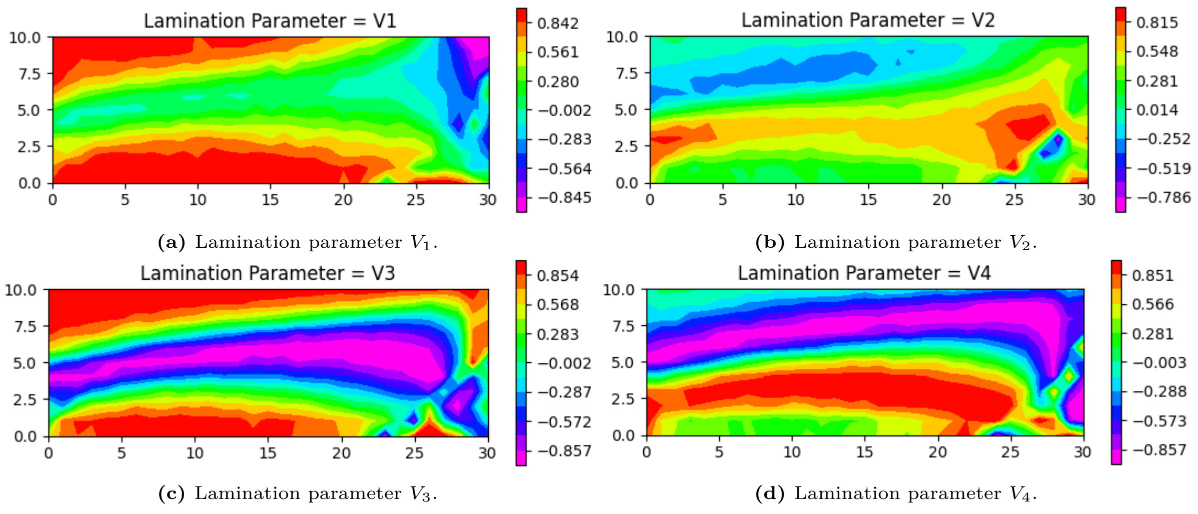
(d) Lamination parameter V_4.

Figure 7.4: Distribution of Lamination parameters obtained using Gradient-based Interior point optimizer. ($a/b = 3$, and 31×11 nodes)

	Non-dimensional compliance	
Method used	$\bar{C}_{gv}$	$\bar{C}_{bv}$
Baseline from reference [62]	0.0296	0.0394
Gradient-based optimizer	0.0235	0.0294
Genetic Algorithm	0.0245	0.0295

Table 7.5: Non-dimensional compliance results obtained from gradient-based optimizer and GA, compared to baseline [62] for $a/b = 3$.

As can be seen in Table 7.5 there are two non-dimensional compliances generated; $\bar{C}_{gv}$ is a general variable stiffness panel with 4 lamination parameters, and $\bar{C}_{bv}$ is a balanced variable stiffness panel with 2 lamination parameters (V_1 and V_3). It must be noted that in [62] the non-dimensional compliance values were erroneously multiplied by 2, as stated by Setoodeh et al in [50]. These errors have been rectified in the results shown in Table 7.5. It can be observed that the gradient-based optimizer provides the best results for a general variable stiffness laminate with a minimal compliance value. The same problem was also evaluated using the GA optimizer from pymoo [63] and it does match up quite well to the result obtained from the gradient-based optimizer as can be seen under the $\bar{C}_{gv}$ column in Table 7.5. The Figure 7.5 shows the LP distributions obtained using GA, and it can be seen that these results seem to be more similar to the results obtained by Setoodeh et al., but are a little jagged and can be attributed to the fact that this optimizer selects different points in the design space to move towards the best, rather than utilizing the convex nature of the design space. These results could possibly improve if the population size is increased, as well as the number of generations, and move towards the global minimum. For the balanced VS laminate, it yields an almost identical result when the non-dimensional compliance is compared with the value obtained using the gradient-based optimizer, however, the distributions for V_1 and V_3 have a few subtle differences (See Appendix D, Figure D.1) that can be attributed to the heuristic nature of GA. It is possible to carry a similar optimization out for the out-of-plane LPs as well, by just replacing the in-plane LPs with out-of-plane ones in the problem formulation. So instead of the A-matrix, the D-matrix will be written in Equation 7.8 which will be written as a function of the out-of-plane LPs.



(a) Lamination parameter V_1.

(b) Lamination parameter V_2.

(c) Lamination parameter V_3.

(d) Lamination parameter V_4.

Figure 7.5: Distribution of Lamination parameters obtained using Genetic Algorithm optimizer. ($a/b = 3$, and 31×11 nodes)

Hence, in this case, study it can be concluded that using Lamination parameters does help make the design space convex and in this case with minimum compliance designs. The design space could be non-convex if the structure under consideration is a complex one (like an aircraft wing) and not just a simple laminate like a plate with in-plane or out-of-plane loads. Further, it was observed how the gradient-based optimizer was able to shine in this case by providing the best result in terms of non-dimensional compliance and can prove to be a vital tool in the design of variable stiffness laminates involving Lamination Parameters.

7.4. Case study 4

7.4.1. Problem description and formulation

Optimization of variable stiffness laminates using Lamination parameters requires only 4 design variables that influence the stiffness properties of an element in a discretized laminate, regardless of the number of layers it has. However once these optimized lamination parameters are obtained, retrieval of the fiber angles associated with the lamination parameters makes the design space increasingly non-convex as the number of layers increases. Retrieval of the fiber angles is a topic that is being studied extensively, and for this case study one of the simpler methods will be utilized to generate the fiber angles for VS laminates. Setoodeh et al. in their work [50] discuss a method for generating curvilinear fiber paths for the manufacturing of VS laminates using tow placement. A part of their work is obtaining the fiber angles at each node of the discretized VS laminate for which the LP distribution is already known. Hence this case study will focus on the retrieval of these fiber angles associated with each node along the VS laminate. Hence the objective function that will be used is the function discussed in chapter 4 given by Equation 4.20 using the least square distance between the optimum lamination parameters and the calculated ones which is based on the fiber angles being optimized. The design variables here are the fiber angles which will be varied such that the LPs are calculated using the relation that relates the LPs to the fiber angles as given in, Equation 4.12. In this case, the bounds for the angles the optimizer can vary the ply angles between are the only constraints considered. A curvature constraint can be added as well to limit how much the fiber angles between adjacent nodes vary, however in this case to test the optimization method finding the minimum value for the objective function is the target.

In this case study the same cantilever plate problem is explored as previously, and the LPs obtained in the previous case study will be the vector of optimum lamination parameters V^*. The plate is considered to be a balanced plate with 2 layers [50], which essentially brings the number of design variables down to just 1. Hence the optimization problem is formulated as,

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & |V - V^*| \\ \text{subject to} \quad & 0 \leqslant \theta \leqslant 90 \end{aligned} \tag{7.11}$$

Where V is the vector of Lamination parameters calculated using the relations Equation 4.12. It can be noted that here the lower limit of θ is $0°$ because the optimization considers a balanced laminate, hence the range has been reduced. Even though the optimization is done for a single design variable, the design space is still non-convex and can be seen in Figure 7.6. The plot shows how the value of the objective function changes at a randomly selected node on the laminate when the balanced angle is changed.

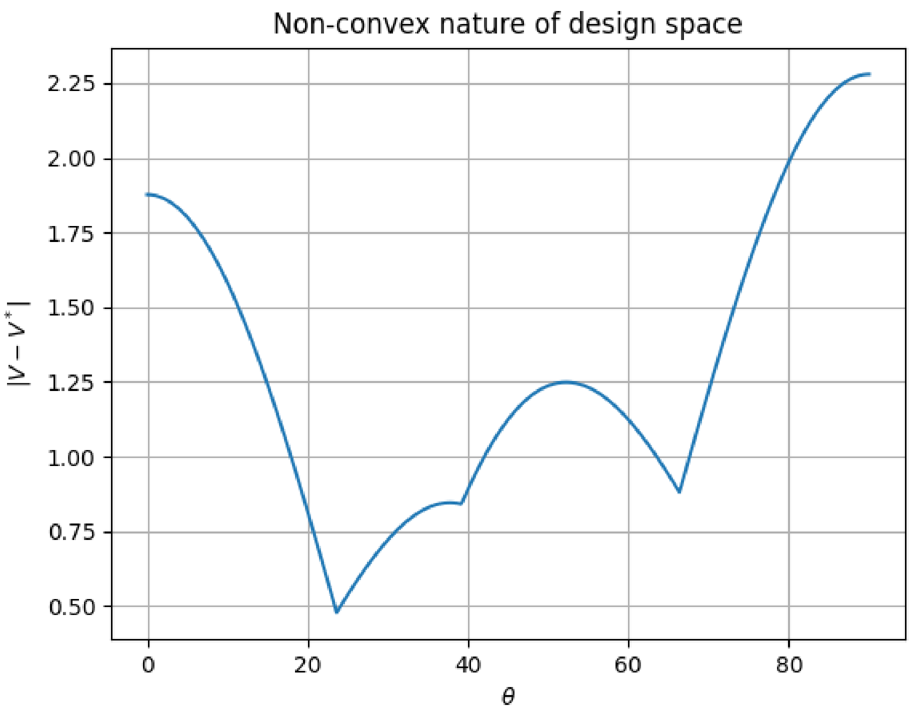
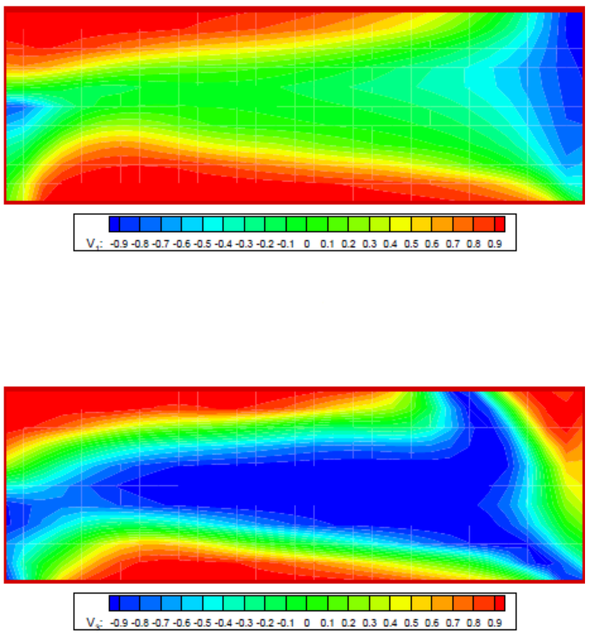


Figure 7.6: Cantilever plate with uniform load [62].

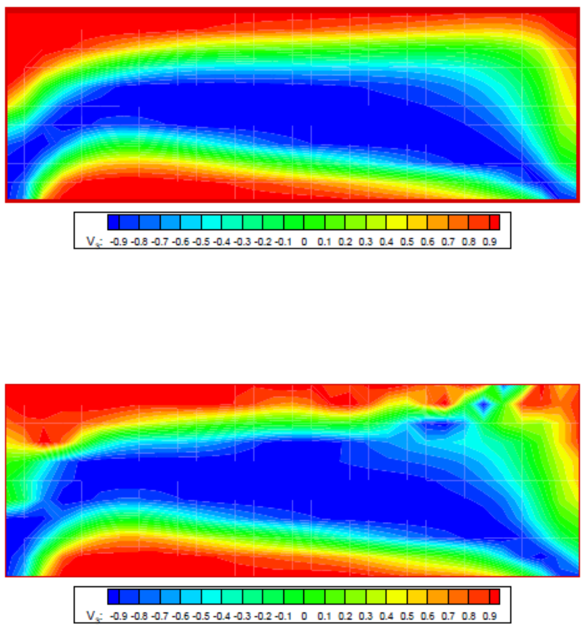
Owing to the non-convex nature of this design space, deflation can be applied in the gradient-based interior optimization algorithm to pick the ply angle that best minimizes the function. The same problem will also be solved using the GA from pymoo [63] to provide a comparison case along with the results obtained by Setoohdeh et al. [50].

7.4.2. Results

As stated previously Setoodeh et al., generated the results for a balanced 2-ply laminate, and for evaluating the results they once a gain computed the non-dimensional compliance as shown in Equation 7.9 using the obtained stacking sequence and compared it to the non-dimensional compliance they obtained after optimization of the LPs as done in the previous case study. The Figure 7.7 shows the distribution of the LPs across the nodes for a balanced laminate. The upper images represent the optimal LP distributions while the images below represent the distribution obtained with the retrieved ply fiber angles. It can be seen that there are a few differences in the optimum and the LPs with retrieved fiber angles which can be attributed to the non-convex nature of the design space. It can be seen in Table 7.6 the $\bar{C}^*$ shows the normalized compliance values with the optimum LPs and $\bar{C}$ shows the ones obtained with the retrieved fiber angles. For the baseline result, Setoodeh et al. were able to achieve fiber angles that were just 4% different from the retrieved fiber angles. It must be noted that here they have utilized a curvature constraint with quite a high upper limit ($\kappa = 3.333m^{-1}$) to obtain the 4% difference. As the maximum curvature reduces, the percentage difference increased because the rotation of the fibers across the nodes were restricted.

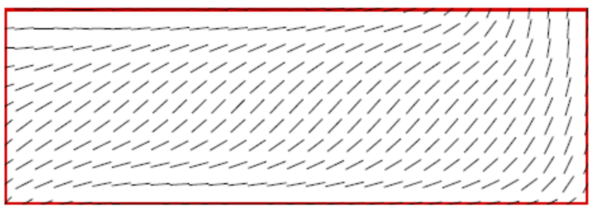


(a) Lamination parameter V_1.

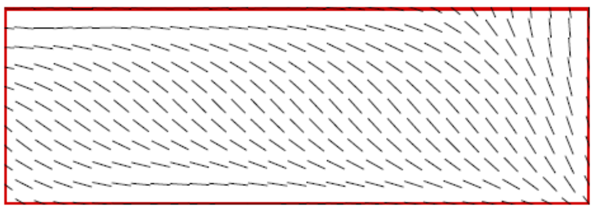


(b) Lamination parameter V_3.

Figure 7.7: Distribution of Lamination parameters obtained by Setoodeh et al. with upper images representing optimal LP values and bottom obtained with retrieved angles. ($a/b = 3$, and 31×11 nodes)



(a) $+\theta$

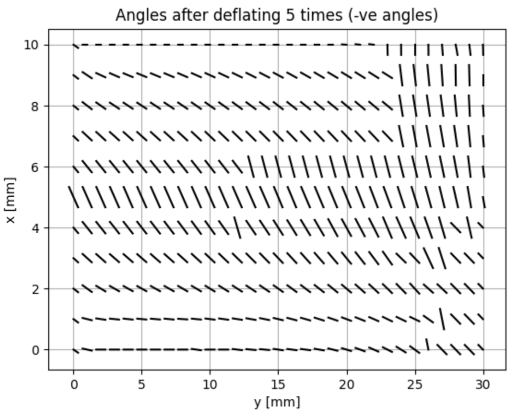


(b) $-\theta$

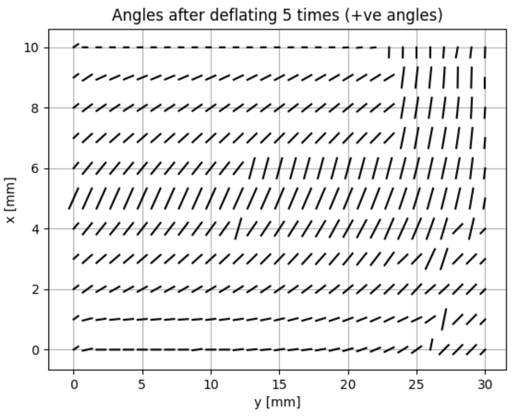
Figure 7.8: Distribution of fiber angles obtained by Setoodeh et al. for a balanced ply layup [50]. ($a/b = 3$, and 31×11 nodes)

Method used	Normalized compliance $\bar{C}^*$	$\bar{C}$	%Difference
Baseline from reference [50]	0.0374	0.0389	4.00
Gradient-based optimizer	0.0294	0.0303	3.06
Genetic Algorithm	0.0295	0.0305	3.38

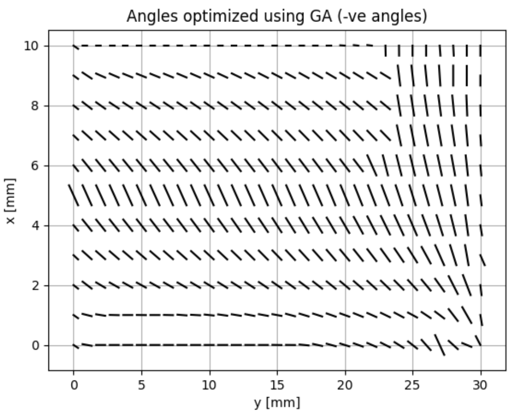
Table 7.6: Normalized compliance obtained with optimum LPs and retrieved stacking sequence.



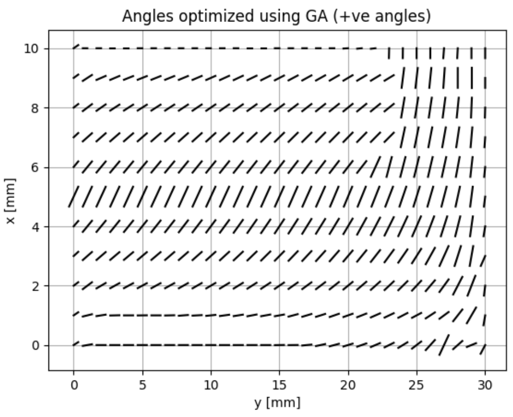
(a) $+\theta$ obtained after deflating 5 times.



(b) $-\theta$ obtained after deflating 5 times.



(c) $+\theta$ obtained with GA.



(d) $-\theta$ obtained with GA.

Figure 7.9: Distribution of fiber angles of balanced laminate, obtained using gradient-based optimization with deflation, and GA, respectively.

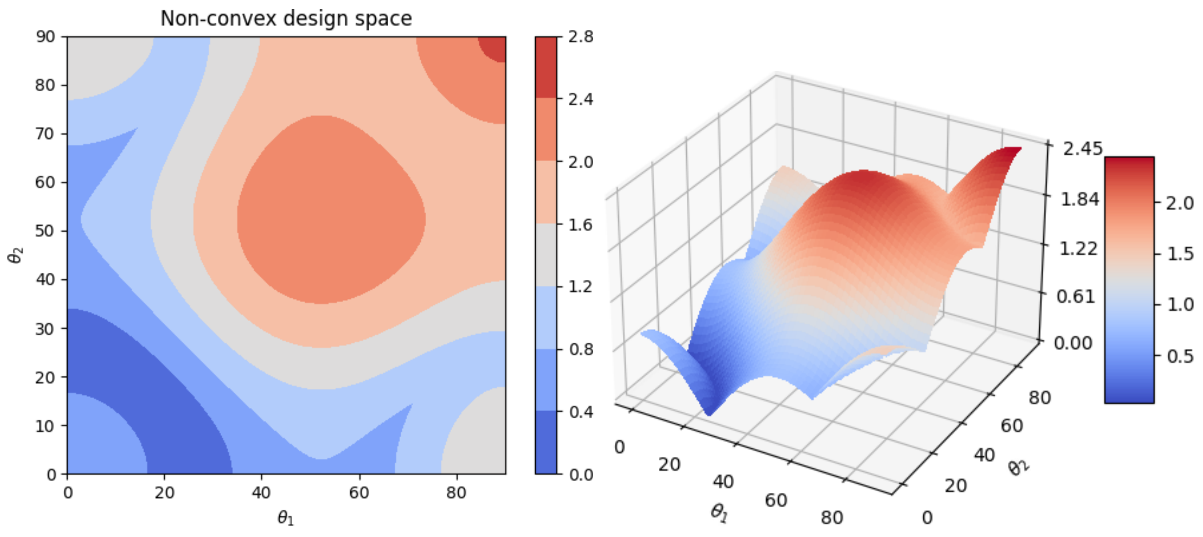
Coming to the results obtained from the gradient-based optimizer and GA; the upper two figures in Figure 7.9 show the positive and negative layers obtained for the balanced Laminate using the developed gradient-based method with deflation. This result was obtained after deflating 5 times (evaluating 6 solutions) at each node of the laminate, and the angle that best minimizes the objective function (Equation 7.11) is selected. In Table 7.6 it can be seen that the percentage difference between the non-dimensional compliances obtained with V^* and V is approx 3.1% apart. This is a slightly better result as compared to the ones obtained with GA and also the baseline from Setoodeh et al. This is a good indication that the global minimum associated with the objective function at the different nodes of the laminate are being achieved. The bottom two figures in Figure 7.9 show the fiber angles obtained with GA, and visually when the two are compared there are quite a few differences. These differences are attributed to the different optimum LP distributions obtained using the two methods. A visual comparison of these LP distributions has been depicted in Figure D.2 and in Figure D.3.

It can be concluded from this case study, that the deflation-based optimization method is providing quite a good result with this simple formulation (Equation 7.11), and has a lot of potential for obtaining the fiber angles associated with multiple layers, i.e. multiple design variables rather than just one.

7.5. Case study 5

7.5.1. Problem description and formulation

The final case study would be building up on the previous one, where the fiber angles along each node for a balanced laminate are retrieved from the optimum lamination parameters from case study 3. However here, the number of plies will be increased, such that the optimizer has to deal with a more complicated design space to obtain the fiber angles stacking at each node that yields LPs that would best match the optimum LPs. The same objective function as Equation 7.11 is used, to locally minimize the objective function across each node. Increasing the number of layers from 2 to 4 for a balanced laminate, increase the design variables to two. Hence this space can still be visualized to actually observe how many local minima there are.



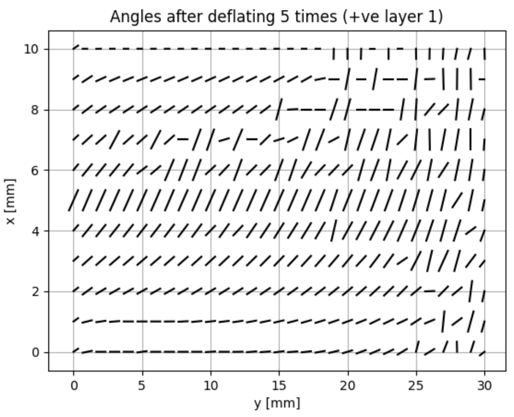
(a) Contour plot of design space with 2 variables. **(b)** 3D plot of design space with 2 variables.

Figure 7.10: Design space plots for a random node along the laminate.

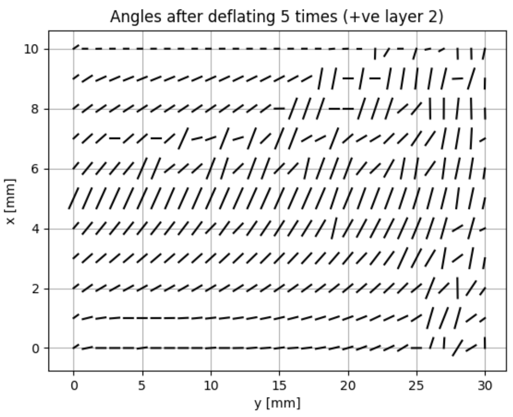
Figure 7.10 shows the plots for the design space as a contour plot and a 3D plot. It can be observed that with this objective function, there is no uniformity associated with how the value of the objective function varies, and there are multiple local minima in the space. Hence once again, the deflation constraint coupled with the gradient-based optimization method will be utilized to obtain the optimal fiber angle distributions for each layer of the laminate. The GA from pymoo will also be used as a comparison case. In this case study, it will be observed how the pecentage difference between the Non-dimensional compliances (as seen in case study 3) varies with the increasing number of layers and how the deflation-based optimization method can cope with the increasing design space.

7.5.2. Results

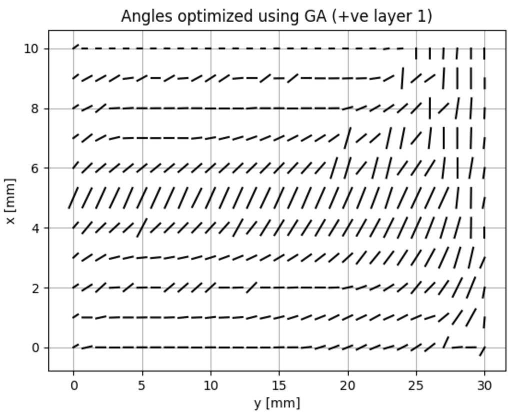
The deflation constraint is set to generate 6 solutions out of which the solution that best minimizes Equation 7.11 is selected. For the case of the 4-layered-balanced laminate, the fiber angle distribution associated with the two design variables has been plotted and can be seen in Figure 7.11. These fiber angles are plotted using the deflation-based method and GA while considering their respective optimum LP distributions and as it can be seen, in both cases, the angles obtained in Layer-1 and Layer-2 are quite different.



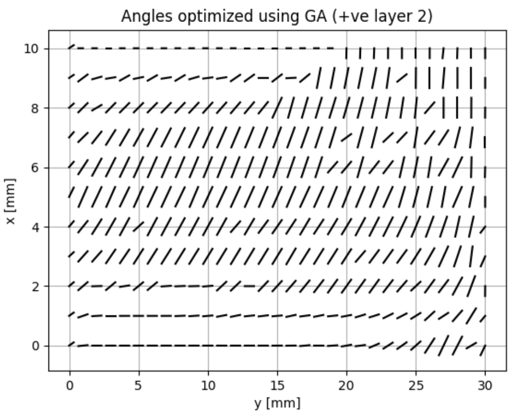
(a) $+\theta_1$ obtained after deflating 5 times.



(b) θ_2 obtained after deflating 5 times.



(c) θ_1 obtained with GA.



(d) θ_2 obtained with GA.

Figure 7.11: Distribution of fiber angles of balanced laminate with 4 layers, obtained using gradient-based optimization with deflation, and GA, respectively.

Number of layers	**Method used**	**Normalized compliance**		**%Difference**
		$\bar{C}^*$	$\bar{C}$	
2	Gradient-based optimizer	0.0294	0.0303	3.06
	Genetic Algorithm	0.0295	0.0305	3.38
4	Gradient-based optimizer	0.02941	0.029486	0.23
	Genetic Algorithm	0.029401	0.029643	0.8
6	Gradient-based optimizer	0.0294	0.02945	0.78
	Genetic Algorithm	0.029401	0.029451	0.37
8	Gradient-based optimizer	0.029412	0.0316	7.4
	Genetic Algorithm	0.029401	0.02946	0.2
10	Gradient-based optimizer	0.02942	0.02962	1.3
	Genetic Algorithm	0.029401	0.02945	0.16

Table 7.7: Normalized compliance obtained with optimum LPs and retrieved stacking sequence for balanced laminates with multiple layers.

Table 7.7 shows the calculated non-dimensional compliances associated with the generated ply fiber distributions and optimal LPs. With the gradient-based optimizer that uses deflation to generate 6 solutions, for 4 and 6 layered balance laminates, the Non-dimensional compliances are a very good match to the optimal values and this could be because the increased number of layers provide more freedom for the laminates to rotate the fibers along the stack to achieve the required LPs. However, for 8 layers, it seems like the obtained non-dimensional compliance values are off quite a bit more by about 7.4% which may be because the optimizer did not obtain a global minimum. Hence, in that case, evaluating more than 6 solutions would need to be done. Coming to the GA, it seems to be performing quite efficiently in almost all the cases and does this with fewer function evaluations as compared to the gradient-based method with deflation.

In this case study it can be observed that the gradient-based method with deflation helps to find the optimal fiber distributions as the design space increases in size and works quite well with the formulation suggested in Equation 7.11. Even though the GA with how it samples new points performs more efficiently in this case, it does not always determine a global minimum. That is where the gradient-based shines, provided a sufficient amount of solutions are determined to scan that particular design space.

7.6. Conclusion

In this chapter, the potential of the gradient-based method coupled with the deflation constraint has been uncovered. With a sufficient number of solutions being determined, and of course selection of the right deflation parameters, the method can achieve results that might as well yield the global minimum in the cases covered above. The method could benefit from some improvements to make computations more efficient, such that it can work faster and this could be explored in future studies to create a method that can rival the speeds with which GA and other optimization frameworks operate.

This page was intentionally left blank.

8 Research review and future recommendations

With the aim of answering the research questions covered in section 1.2, the following sections are presented as highlights of this thesis and points of discussion. At the end of each section the recommendations for future work that stem from these discussions are proposed.

8.1. Gradient-based optimization algorithm

The methodology of the gradient-based algorithm developed for this thesis is covered in chapter 5 where the developed method is tested and verified against various examples obtained from a reference [6]. The testing also provides insight into the parameters associated with the method and how the variation of said parameters affects the obtained solutions. It was concluded that the optimal set of required parameters based on the minimization of evaluation time or the minimization of the number of iterations required to obtain a minimum, is quite difficult to predict as it changes from problem to problem. It would require a few runs to gauge the effectiveness of these parameters in a certain design space and provide appropriate parameters to get its performance up to expectations.

8.1.1. Inclusion of constraints

The chapter 2 provides a great overview of gradient-based optimization by explaining certain concepts and methods referenced from the literature. The chapter covers how constraints can be included in an optimization algorithm by implementing the Lagrangian and Lagrange multipliers followed by the derivation of the optimality conditions that need to be satisfied to yield a search direction that points in the direction of descent.

8.1.2. Choice of gradient-based method

A brief overview of penalty methods for constrained optimization and SQP, a method for equality constrained optimization which is used to develop the optimization algorithm in this thesis is also given in this chapter. For problems with inequality constraints or a combination of equality and equality constraints, the formulation of the interior-point optimization method is implemented into the optimization algorithm. This is because interior point methods are better formulated when it comes to inequality constraints, as the formulation contains slack variables which serve as an indicator of whether the optimizer is in a feasible region or not. Whereas with SQP the constraints are evaluated at every iteration to establish a working set, which contains the active constraints and these are included in the KKT system of equations [6]. Hence this proves to be quite inefficient.

8.1.3. Line-search method

The chapter 2 covers a section about Line-search for Optimization Algorithms. These methods are required to help the optimizer, not overstep in a particular search direction such that prevents the optimizer from zig-zagging till a minimum is reached and improves efficiency. Line search algorithms determine the amount that needs to be stepped in the particular descent direction and the line search algorithm implemented in the developed optimization algorithm is a back-tracking line search that works on satisfying the condition of sufficient decrease. The strong Wolfe conditions prove to be excessive as the time taken to evaluate the parameters required in evaluating these conditions scale up the total time taken, as compared to the back-tracking method. The difference increases with an increase in the number of design variables.

8.1.4. Automatic differentiation

AD is the differentiation technique presented in chapter 2 which is based on a numerical version of the chain rule and is used to calculate the gradients utilized for the developed optimizer. To apply this method, a pre-existing Python module is utilized (Autograd [13]) and is set to operate in reverse mode which proves to be efficient in the case of an increasing number of design variables, but a lower number of output functions that depend on the input design variables. A disadvantage to this module is that it operates with the tensor data type, which does not apply to all other libraries in Python. Another disadvantage would be that it does not support gradients of non-smooth functions like (max., min., etc.). But overall the gradients are estimated quite accurately.

8.1.5. Future recommendations

For future work in the direction of optimization methodology, efforts can be made to compare the performance of different line-search methods, especially ones that have improved performance by the advent of heuristics in the method [60]. This would improve the convergence behavior of gradient-based optimizers and result in minimal required iterations.

Another suggestion would be to work on an AD method that can operate with data types that are common with other Python libraries to enable the seamless implementation of AD methods in optimization. Also, the support for gradients of non-smooth functions would be an added advantage.

8.2. Deflation method

8.2.1. Concept and parameters involved in deflation

The chapter 3 covers the basic concepts involved in deflation and goes over the history of how the method has evolved. It covers how the concept can be applied to polynomial deflation, followed by the deflation of a system of non-linear equations.

8.2.2. Deflation constraint

With these basics cleared, chapter 6 introduces a traditional approach to applying deflation to an optimization method, where the KKT conditions expressed as residuals can be multiplied by the deflation matrix M to get the new deflated system of equations. But claims of Tarek et al. [23] suggest that this introduces numerical difficulties associated with the Hessian of the Lagrangian function. Hence it is a possible motivation as to why Papadopoulos et al. [17], created a new method for applying deflation with a primal-dual interior point optimization algorithm. Hence the need for a deflation method that can be applied without modifying the existing optimization framework is fulfilled by the deflation constraint.

The deflation constraint is an approach originally introduced by Tarek et al. [23] and lay the foundation for the approach. The work covered in chapter 6 however, is an original and novel formulation of the approach. The results obtained by the approach developed in the present work perform satisfactorily by locating if not all, almost all the minima as shown in the test case depicted by Figure 6.7a. Whereas that is not the case for the approach presented in [23] which fails to deflate after having found a certain number of solutions. One very important and also newly introduced parameter in the present work, the radius of action r, controls the radius around which the deflation constraint acts and plays a vital role. Too large a value can lead to some significant minima being engulfed, hence smaller values of r are recommended for the discovery of all significant minima, even if more function evaluations are needed. Another outcome of this chapter is that it verifies the original hypothesis that a deflated gradient-based method can perform as a global optimizer.

8.2.3. Future recommendations

The deflation constraint method opens up a lot of possibilities for future developments. The current formulation could on its own be improved in two ways. One could be to have the value of the radius of action r vary dynamically where each unique minima has its value of r such that when the optimizer reaches that minimum, the unique r becomes the metric of distance measure. This approach can be quite straightforward while dealing with 2D problems, but higher dimensions would require further study. The second way to improve the formulation would be to modify the formulation further and remove the dependence of r such that it deflates the n-dimensional portion associated with a minimum which is equivalent to a valley in 2D.

Another possible improvement that can be achieved computationally is the parallelization of deflation. The current static (one optimization problem at a time) nature of the deflated gradient-based method developed in this thesis can be adapted to become dynamic, which in essence would be a sequence of optimization problems solved simultaneously while the solver makes active decisions at different time instances, based on the information that becomes available as time progresses. With parallelization, for instance, with a more powerful computational device, there could be three or more deflated-gradient-based optimizers with randomly dispersed starting points in a design space running parallel. As each of the optimizers finds minima, the list of found minima gets updated and the other solvers have the information for the deflation constraint to deflate away from already found minima. This idea would greatly enhance the ability for global optimization while utilizing the deflation constraint being applied to almost any optimization method compatible with constrained optimization.

8.3. Optimization of composite structures

The methodology proposed in this thesis is applied to optimization case studies involving composite structures and is covered in chapter 7.

8.3.1. Constant stiffness stacking angle optimization

In the first case study stacking sequence optimization of a balanced and symmetric composite laminate for critical buckling under bi-axial load was explored. In this case, the developed methodology was compared against a free-to-use Genetic algorithm (GA) optimizer. The baseline was set by the results obtained in literature [61]. Since this optimization involved optimizing layers with discrete angles $(0, \pm 45, 90)$ and not continuous, the continuous angles obtained by the deflated gradient-based solver were converted into these discrete values after optimization, and this resulted in buckling loads that were below what was achieved in the baseline. The free-to-use GA however performed the best. Hence the proposed method as predicted performed poorly in such a discrete-based optimization problem.

In the next case study, the same problem was explored but with the main objective being the volume of the laminate while the buckling load was considered a target constraint. For the weight minimization, here even the ply thicknesses were considered as a variable with the stacking sequence, essentially doubling the design variables. After observing the previous case of how the deflated gradient method performed poorly, the free-to-use GA was utilized and this time the deflation constraint was applied to it. This resulted in quite a good test, where it was verified that the deflation constraint can be used with any off-the-shelf optimization method, and also the deflated GA generated superior solutions as compared to non-deflated GA.

8.3.2. Variable stiffness laminates

In chapter 4 the literature study associated variable stiffness laminates are covered. It shares brief information about the various available manufacturing techniques, and the various defects that can be encountered during manufacturing followed by two known methods for parameterizing variable stiffness. The first method is direct fiber path parameterization and in this method, the fiber paths are parameterized by using various interpolation functions, modeled as Bezier curves, etc. The second method is to model the stiffness properties using 12 parameters known as lamination parameters. 4 Lamination parameters are associated with each, in-plane, coupling, and bending stiffness. The advantage offered by this parameterization is that the design space occupied by LPs is convex which is a major advantage as compared to the highly non-convex space of fiber path parameterization. However, one major challenge faced with optimizing lamination parameters is that the retrieval of fiber angles is quite tedious, as it would once again lead to non-convex optimization to help match the obtained LPs and the LPs corresponding to the optimized fiber angles.

In chapter 7, the third case study covered explores the optimization of Lamination parameters for minimizing the compliance of a cantilever plate with uniform loading. The gradient-based optimizer operates without deflation in this case as it takes advantage of the convex design space to obtain the 4 optimized in-plane lamination parameters. The result provides a good match with literature [62] while yielding a slightly better result when the non-dimensional compliance is compared. The GA is also used as a comparison case but does not improve over the result obtained by the gradient-based method.

8.3.3. Fiber angle retrieval

In chapter 7, after optimization of the Lamination Parameters, a simple objective function is utilized to retrieve the fiber angles. Literature [67] provides a baseline obtained fiber layup for a 2-ply, balanced laminate. Which is essentially only 1 design variable. However, the design space proved to be a non-convex one owing to the objective function. The baseline is set by the reference literature, and the optimization is carried out by utilizing the deflated-gradient-based optimizer and the GA. The deflated optimizer yields the best result when non-dimensional compliance is compared. Hence this method can be used for fiber angle retrieval.

Another case study is performed but for this, there is no established literature available. The same objective function is utilized for obtaining the fiber angle distribution for balanced laminates with a different number of layers. This case study proves that the deflated gradient-based method is quite robust as it generates results that are comparable if not better than GA.

8.3.4. Future recommendations

The case studies covered in this thesis form only the basis for the capabilities of this method. Future studies could implement this deflation constraint method for the global optimization of even more complex models for VS structures that have a variety of loads acting on them which would result in highly non-convex design spaces. Manufacturability can also be considered in this case by applying fiber curvature constraints, and other manufacturing constraints to the optimization problem. The Ph.D. thesis of Peeters [34] covers this topic in great depth and would serve as a great starting point for the implementation of manufacturing constraints associated with VS structures

For VS laminates, new methods could be explored for fiber angle retrieval from optimized Lamination parameters with an emphasis on the development of different and better-formulated objective functions.

This page was intentionally left blank.

9 Conclusion

Based on the research questions covered in section 1.2, the primary research objective of this thesis is formulated as follows:

Primary Research Objective

To create a deflated gradient-based optimization method that implements a deflation method that can be used for global optimization of the non-convex design spaces associated with composite structural design problems.

This thesis project achieves the stated research objective by developing a methodology for achieving deflation not only limited to the originally targeted gradient-based optimization algorithm, but also applicable to other optimization algorithms such as genetic algorithms, or virtually any other optimization algorithm that is compatible with constraints.

As gradient-based optimizers are known for their ability to recover local minima, a gradient-based method was selected for the application of deflation to aid in facilitating gradient-based optimizers to transition from being local optimizers to becoming global optimizers. The developed gradient-based optimization algorithm is capable of optimizing constrained and unconstrained problems.

The deflation method herein developed is deflation based on an inequality constraint, which can be applied to the optimization algorithm to force the optimizer away from existing minima, enabling the discovery of new minima in a non-convex design space. The original deflation formulation herein presented is the main contribution of this thesis.

The developed methodology has been implemented in multiple case studies that support the optimization of composite structures. The case studies involve the optimization of stacking sequence associated with constant stiffness laminates, followed by the optimization of lamination parameters of a variable stiffness laminate, and the optimization of the spatial fiber orientation angles to match the optimized lamination parameters.

This page was intentionally left blank.

References

[1] C. Kassapoglou, *Design and Analysis of Composite Structures.* Oxford, UK: John Wiley & Sons Ltd, May 2013, ISBN: 9781118536933. DOI: 10.1002/9781118536933.

[2] J. M. Van Campen, C. Kassapoglou, and Z. Gürdal, "Generating realistic laminate fiber angle distributions for optimal variable stiffness laminates," *Composites Part B: Engineering*, vol. 43, no. 2, pp. 354–360, Mar. 2012, ISSN: 13598368. DOI: 10.1016/j.compositesb.2011.10.014.

[3] M. A. Albazzan, R. Harik, B. F. Tatting, and Z. Gürdal, *Efficient design optimization of nonconventional laminated composites using lamination parameters: A state of the art*, Feb. 2019. DOI: 10.1016/j.compstruct.2018.10.095.

[4] S. Kucherenko and Y. Sytsko, "Application of Deterministic Low-Discrepancy Sequences in Global Optimization," *Computational Optimization and Applications*, vol. 30, no. 3, pp. 297–318, Mar. 2005, ISSN: 0926-6003. DOI: 10.1007/s10589-005-4615-1.

[5] P. Kaelo and M. M. Ali, "Some Variants of the Controlled Random Search Algorithm for Global Optimization," *Journal of Optimization Theory and Applications*, vol. 130, no. 2, pp. 253–264, Dec. 2006, ISSN: 0022-3239. DOI: 10.1007/s10957-006-9101-0.

[6] J. R. R. A. Martins and S. A. (A. Ning, *Engineering design optimization*, p. 637, ISBN: 9781108833417.

[7] X. He, J. Li, C. A. Mader, A. Yildirim, and J. R. Martins, "Robust aerodynamic shape optimization—From a circle to an airfoil," *Aerospace Science and Technology*, vol. 87, pp. 48–61, Apr. 2019, ISSN: 12709638. DOI: 10.1016/j.ast.2019.01.051.

[8] Z. Lyu, G. K. W. Kenway, and J. R. R. A. Martins, "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," *AIAA Journal*, vol. 53, no. 4, pp. 968–985, Apr. 2015, ISSN: 0001-1452. DOI: 10.2514/1.J053318.

[9] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016.

[10] A. Rakhlin, O. Shamir, and K. Sridharan, "Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization," Sep. 2011.

[11] A. Griewank and A. Walther, *Evaluating Derivatives.* Society for Industrial and Applied Mathematics, Jan. 2008, ISBN: 978-0-89871-659-7. DOI: 10.1137/1.9780898717761.

[12] U. Naumann, *The Art of Differentiating Computer Programs.* Society for Industrial and Applied Mathematics, Jan. 2011, ISBN: 978-1-61197-206-1. DOI: 10.1137/1.9781611972078.

[13] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in PyTorch," Tech. Rep.

[14] Y. Zhu, B. Guillemat, and O. Vitrac, "Rational Design of Packaging: Toward Safer and Ecodesigned Food Packaging Systems," *Frontiers in Chemistry*, vol. 7, May 2019, ISSN: 2296-2646. DOI: 10.3389/fchem.2019.00349.

[15] A. H. G. Rinnooy Kan and G. T. Timmer, "Stochastic global optimization methods part I: Clustering methods," *Mathematical Programming*, vol. 39, no. 1, pp. 27–56, Sep. 1987, ISSN: 0025-5610. DOI: 10.1007/BF02592070.

[16] A. Arnoud, F. Guvenen, and T. Kleineberg, "Benchmarking Global Optimizers," National Bureau of Economic Research, Cambridge, MA, Tech. Rep., Oct. 2019. DOI: 10.3386/w26340.

[17] I. P. A. Papadopoulos, P. E. Farrell, and T. M. Surowiec, "Computing multiple solutions of topology optimization problems," Apr. 2020. DOI: https://doi.org/10.48550/arXiv.2004.11797.

[18] J. Xia, P. E. Farrell, and S. G. Castro, "Nonlinear bifurcation analysis of stiffener profiles via deflation techniques," *Thin-Walled Structures*, vol. 149, p. 106 662, Apr. 2020, ISSN: 02638231. DOI: 10.1016/j.tws.2020.106662.

[19] J. H. Wilkinson, *Rounding errors in algebraic processes.* 1994.

[20] P. E. Farrell, Á. Birkisson, and S. W. Funke, "Deflation Techniques for Finding Distinct Solutions of Nonlinear Partial Differential Equations," *SIAM Journal on Scientific Computing*, vol. 37, no. 4, A2026–A2045, Jan. 2015, ISSN: 1064-8275. DOI: 10.1137/140984798.

[21] P. Deuflhard, *Newton Methods for Nonlinear Problems.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 35, ISBN: 978-3-642-23898-7. DOI: 10.1007/978-3-642-23899-4.

[22] K. M. Brown and W. B. Gearhart, "Deflation techniques for the calculation of further solutions of a nonlinear system," *Numerische Mathematik*, vol. 16, no. 4, pp. 334–342, Jan. 1971, ISSN: 0029-599X. DOI: 10.1007/BF02165004.

[23] M. Tarek and Y. Huang, *Simplifying deflation for non-convex optimization with applications in Bayesian inference and topology optimization*, 2022.

[24] P. E. Farrell, M. Croci, and T. M. Surowiec, "Deflation for semismooth equations," *Optimization Methods and Software*, vol. 35, no. 6, pp. 1248–1271, Nov. 2020, ISSN: 1055-6788. DOI: 10.1080/10556788.2019.1613655.

[25] M. W. Hyer, P. R. Investigator F Charette, and D. Taylor, "INNOVATIVE DESIGN OF COMPOSITE STRUCTURES: THE USE OF CURVILINEAR FIBER FORMAT IN COMPOSITE STRUCTURE DESIGN," Tech. Rep., 1990.

[26] M. Hyer and H. Lee, "The use of curvilinear fiber format to improve buckling resistance of composite plates with central circular holes," *Composite Structures*, vol. 18, no. 3, pp. 239–261, Jan. 1991, ISSN: 02638223. DOI: 10.1016/0263-8223(91)90035-W.

[27] S. B. Biggers and S. Srinivasan, "Compression Buckling Response of Tailored Rectangular Composite Plates," *AIAA Journal*, vol. 31, no. 3, pp. 590–596, Mar. 1993, ISSN: 0001-1452. DOI: 10.2514/3.61543.

[28] A. Cherouat, H. Borouchaki, J.-L. Billoët, and *. .-. Borouchaki, "Geometrical and mechanical drap-ing of composite fabric. Revue Européenne des Éléments Finis," pp. 693–707, 2005. DOI: 10.3166/reef.14.693-707{\"{i}}.

[29] E. Kunze, S. Galkin, R. Böhm, M. Gude, and L. Kärger, "The Impact of Draping Effects on the Stiffness and Failure Behavior of Unidirectional Non-Crimp Fabric Fiber Reinforced Composites," *Materials*, vol. 13, no. 13, p. 2959, Jul. 2020, ISSN: 1996-1944. DOI: 10.3390/ma13132959.

[30] A. Crosky, C. Grant, D. Kelly, X. Legrand, and G. Pearce, "Fibre placement processes for composites manufacture," in *Advances in Composites Manufacturing and Process Design*, Elsevier, 2015, pp. 79–92. DOI: 10.1016/B978-1-78242-307-2.00004-X.

[31] B. C. Kim, K. Potter, and P. M. Weaver, "Continuous tow shearing for manufacturing variable angle tow composites," *Composites Part A: Applied Science and Manufacturing*, vol. 43, no. 8, pp. 1347–1356, Aug. 2012, ISSN: 1359835X. DOI: 10.1016/j.compositesa.2012.02.024.

[32] W. T. Nugroho, Y. Dong, and A. Pramanik, "3D printing composite materials: A comprehensive review," in *Composite Materials*, Elsevier, 2021, pp. 65–115. DOI: 10.1016/b978-0-12-820512-9.00013-7.

[33] A. Brasington, C. Sacco, J. Halbritter, R. Wehbe, and R. Harik, "Automated fiber placement: A review of history, current technologies, and future paths forward," *Composites Part C: Open Access*, vol. 6, p. 100 182, Oct. 2021, ISSN: 26666820. DOI: 10.1016/j.jcomc.2021.100182.

[34] D. Peeters, "Design Optimisation of Practical Variable Stiffness and Thickness Laminates," DOI: 10.4233/uuid:a07ea6a4-be73-42a6-89b5-e92d99bb6256.

[35] K. Croft, L. Lessard, D. Pasini, M. Hojjati, J. Chen, and A. Yousefpour, "Experimental study of the effect of automated fiber placement induced defects on performance of composite laminates," *Composites Part A: Applied Science and Manufacturing*, vol. 42, no. 5, pp. 484–491, May 2011, ISSN: 1359835X. DOI: 10.1016/j.compositesa.2011.01.007.

[36] J. Chen, T. Chen-Keat, M. Hojjati, A. Vallee, M.-A. Octeau, and A. Yousefpour, "Impact of layup rate on the quality of fiber steering/cut-restart in automated fiber placement processes," *Science and Engineering of Composite Materials*, vol. 22, no. 2, pp. 165–173, Mar. 2015, ISSN: 2191-0359. DOI: 10.1515/secm-2013-0257.

[37] K. Fayazbakhsh, M. Arian Nik, D. Pasini, and L. Lessard, “Defect layer method to capture effect of gaps and overlaps in variable stiffness laminates made by Automated Fiber Placement,” *Composite Structures*, vol. 97, pp. 245–251, Mar. 2013, ISSN: 02638223. DOI: 10.1016/j.compstruct.2012.10.031.

[38] M. Arian Nik, K. Fayazbakhsh, D. Pasini, and L. Lessard, “Optimization of variable stiffness composites with embedded defects induced by Automated Fiber Placement,” *Composite Structures*, vol. 107, no. 1, pp. 160–166, Jan. 2014, ISSN: 02638223. DOI: 10.1016/j.compstruct.2013.07.059.

[39] V. Mishra, D. M. Peeters, and M. M. Abdalla, “Stiffness and buckling analysis of variable stiffness laminates including the effect of automated fibre placement defects,” *Composite Structures*, vol. 226, Oct. 2019, ISSN: 02638223. DOI: 10.1016/j.compstruct.2019.111233.

[40] S. G. Castro, M. V. Donadon, and T. A. Guimarães, “ES-PIM applied to buckling of variable angle tow laminates,” *Composite Structures*, vol. 209, pp. 67–78, Feb. 2019, ISSN: 02638223. DOI: 10.1016/j.compstruct.2018.10.058.

[41] B. C. Kim, P. M. Weaver, and K. Potter, “Computer aided modelling of variable angle tow composites manufactured by continuous tow shearing,” *Composite Structures*, vol. 129, pp. 256–267, Oct. 2015, ISSN: 02638223. DOI: 10.1016/j.compstruct.2015.04.012.

[42] Z. Gürdal, B. F. Tatting, and C. K. Wu, “Variable stiffness composite panels: Effects of stiffness variation on the in-plane and buckling response,” *Composites Part A: Applied Science and Manufacturing*, vol. 39, no. 5, pp. 911–922, May 2008, ISSN: 1359835X. DOI: 10.1016/j.compositesa.2007.11.015.

[43] T. A. Guimarães, S. G. Castro, D. A. Rade, and C. E. Cesnik, “Panel flutter analysis and optimization of composite tow steered plates,” in *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2017*, American Institute of Aeronautics and Astronautics Inc, AIAA, 2017, ISBN: 9781624104534. DOI: 10.2514/6.2017-1118.

[44] Z. Wu, P. M. Weaver, G. Raju, and B. Chul Kim, “Buckling analysis and optimisation of variable angle tow composite plates,” *Thin-Walled Structures*, vol. 60, pp. 163–172, Nov. 2012, ISSN: 02638231. DOI: 10.1016/j.tws.2012.07.008.

[45] L. Parnas, S. Oral, and Ü. Ceyhan, “Optimum design of composite structures with curved fiber courses,” *Composites Science and Technology*, vol. 63, no. 7, pp. 1071–1082, 2003, ISSN: 02663538. DOI: 10.1016/S0266-3538(02)00312-3.

[46] A. Alhajahmad, M. M. Abdallah, and Z. Gürdal, “Design tailoring for pressure pillowing using tow-placed steered fibers,” *Journal of Aircraft*, vol. 45, no. 2, pp. 630–640, 2008, ISSN: 15333868. DOI: 10.2514/1.32676.

[47] H. Ghiasi, K. Fayazbakhsh, D. Pasini, and L. Lessard, *Optimum stacking sequence design of composite materials Part II: Variable stiffness design*, Dec. 2010. DOI: 10.1016/j.compstruct.2010.06.001.

[48] S. W. Tsai and N. J. Pagano, “INVARIANT PROPERTIES OF COMPOSITE MATERIALS,” Tech. Rep., 1968.

[49] S. W. Tsai and H. T. Hahn, *Introduction to Composite Materials*. Routledge, May 2018, ISBN: 9780203750148. DOI: 10.1201/9780203750148.

[50] S. Setoodeh, A. W. Blom, M. M. Abdalla, and Z. Gürdal, “Generating curvilinear fiber paths from lamination parameters distribution,” in *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, vol. 5, American Institute of Aeronautics and Astronautics Inc., 2006, pp. 3440–3452, ISBN: 1563478080. DOI: 10.2514/6.2006-1875.

[51] J. Dillinger, “Static Aeroelastic Optimization of Composite Wings with Variable Stiffness Laminates,” Tech. Rep.

[52] J. Grenestedt and P. Gudmundson, “Layup Optimization of Composite Material Structures,” in *Optimal Design with Advanced Materials*, Elsevier, 1993, pp. 311–336. DOI: 10.1016/b978-0-444-89869-2.50027-5.

[53] M. Miki, “Design of Laminated Fibrous Composite Plates with Required Flexural Stiffness,” in *Recent Advances in Composites in the United States and Japan*, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959: ASTM International, pp. 387–387. DOI: 10.1520/STP32802S.

[54] M. Miki and Y. Sugiyama, “Optimum design of laminated composite plates using lamination parameters,” *AIAA Journal*, vol. 31, no. 5, pp. 921–922, 1993, ISSN: 00011452. DOI: 10.2514/3.49033.

[55] J. M. Van Campen, “Optimum lay-up design of variable stiffness composite structures,” Tech. Rep., 2011.

[56] H. Fukunaga and H. Sekine, “Stiffness design method of symmetric laminates using lamination parameters,” *AIAA Journal*, vol. 30, no. 11, pp. 2791–2793, 1992, ISSN: 00011452. DOI: 10.2514/3.11304.

[57] Z. Wu, G. Raju, and P. M. Weaverz, “Feasible region of lamination parameters for optimization of variable angle tow (VAT) composite plates,” in *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2013, ISBN: 9781624102233. DOI: 10.2514/6.2013-1481.

[58] Springer, “Springer Complexity,” Tech. Rep.

[59] A. Ashok, “Bayesian Optimization for Lightweight Design of Variable Stiffness Composite Cylinders,” Tech. Rep., 2021.

[60] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, May 2006, ISSN: 00255610. DOI: 10.1007/s10107-004-0559-y.

[61] R. Le Riche and R. T. Haftka, “Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm,” *AIAA Journal*, vol. 31, no. 5, pp. 951–956, 1993, ISSN: 00011452. DOI: 10.2514/3.11710.

[62] S. Setoodeh, M. M. Abdalla, and Z. Gürdal, “Design of variable-stiffness laminates using lamination parameters,” *Composites Part B: Engineering*, vol. 37, no. 4-5, pp. 301–309, Jun. 2006, ISSN: 13598368. DOI: 10.1016/j.compositesb.2005.12.001.

[63] J. Blank and K. Deb, “Pymoo: Multi-Objective Optimization in Python,” *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990567.

[64] B. Schläpfer and G. Kress, “A sensitivity-based parameterization concept for the automated design and placement of reinforcement doublers,” *Composite Structures*, vol. 94, no. 3, pp. 896–903, Feb. 2012, ISSN: 02638223. DOI: 10.1016/j.compstruct.2011.08.034.

[65] S. Castro, *pyfe3d*, 2023. DOI: https://doi.org/10.5281/zenodo.7703876.

[66] I. Koutromanos, *Fundamentals of Finite Element Analysis: Linear Finite Element Analysis.* Wiley, 2018.

[67] S. Setoodeh, M. M. Abdalla, S. T. IJsselmuiden, and Z. Gürdal, “Design of variable-stiffness composite panels for maximum buckling load,” *Composite Structures*, vol. 87, no. 1, pp. 109–117, Jan. 2009, ISSN: 02638223. DOI: 10.1016/j.compstruct.2008.01.008.

A
Literature study

A.1. Optimization

A.1.1. Gradient-based optimization

Here an example that helps visualize the gradient of a function in 2D is presented. Consider the following function with two variables [6]:

$$f(x_1, x_2) = x_1^3 + 2x_1x_2^2 - x_2^3 - 20x_1 \tag{A.1}$$

The gradient vector for this function can be calculated as follows,

$$\nabla f(x_1, x_2) = \begin{bmatrix} 3x_1^2 + 2x_2^2 - 20 \\ 4x_1x_2 - 3x_2^2 \end{bmatrix} \tag{A.2}$$

Figure A.1 shows how the gradient vectors at different points are oriented and it can be seen that they point along the direction of steepest local increase.

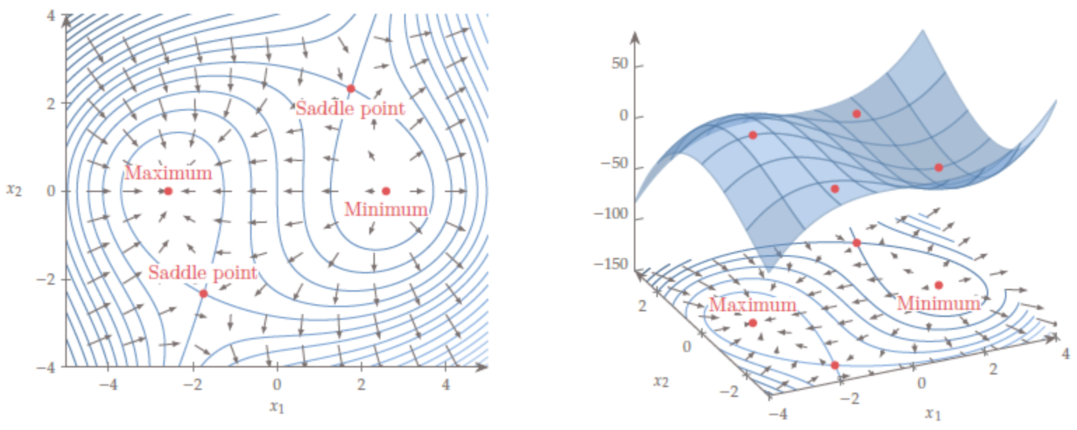


Figure A.1: Gradient vector field showing how the gradients point towards maxima and away from minima [6].

For the same function (Equation A.1) the Hessian can be obtained by differentiating the gradient with respect to the two coordinate directions to obtain,

$$H(x_1, x_2) = \begin{bmatrix} 6x_1 & 4x_2 \\ 4x_2 & 4x_1 - 6x_2 \end{bmatrix} \tag{A.3}$$

The variation of the Hessian can be visualized as shown in Figure A.2 by plotting the principal curvatures at different points. Here the points marked as saddle points can be described as a point at which the principle curvatures contain both, positive and negative curvature, hence yielding a shape that resembles a saddle.

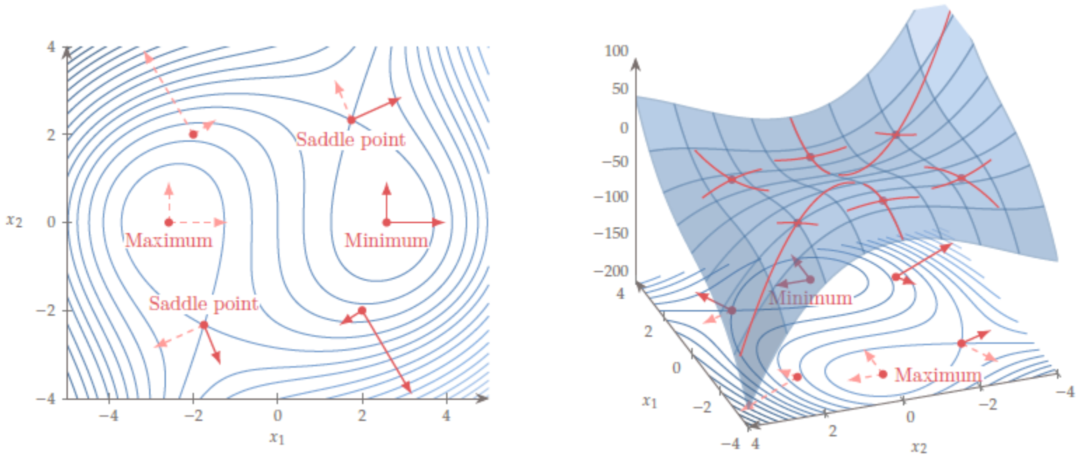


Figure A.2: The directions of principal curvature and associated magnitude variation. The solid lines represent positive curvature while the dashed represent negative curvature [6].

A.2. Variable stiffness Laminates

This appendix displays the feasible regions for the four lamination parameters studied by Wu et al. [57] which are described using Equation 4.19.

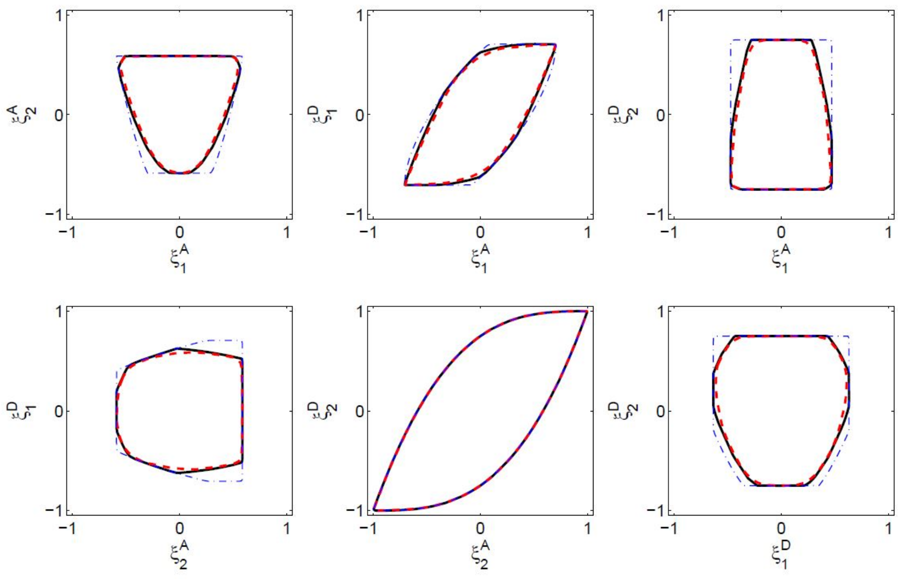


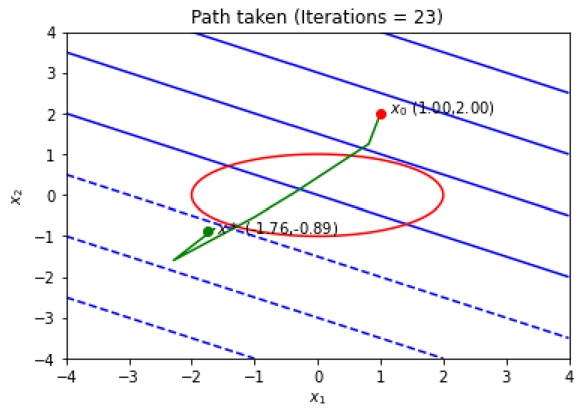
Figure A.3: Feasible domain for four in-plane lamination parameters [57].

B

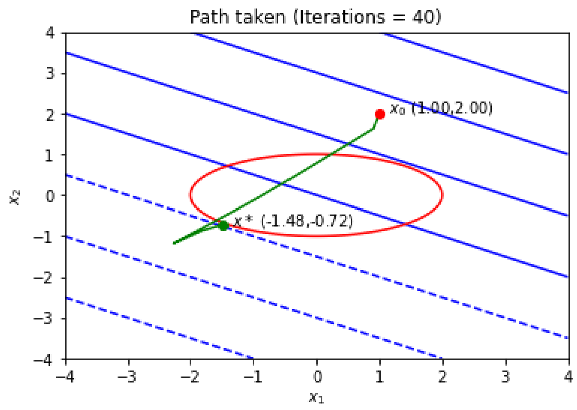
Interior point optimizer

B.1. Finding the right parameters

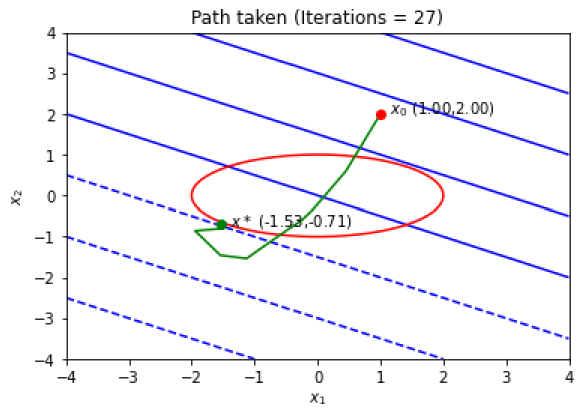
The penalty parameter μ_p of the merit function and the barrier parameter μ_b are quite vital contributors to the number of iterations the optimizer takes to reach the minimum, and also the path it follows. The variation in the number of iterations and path taken can be observed in the figure below,



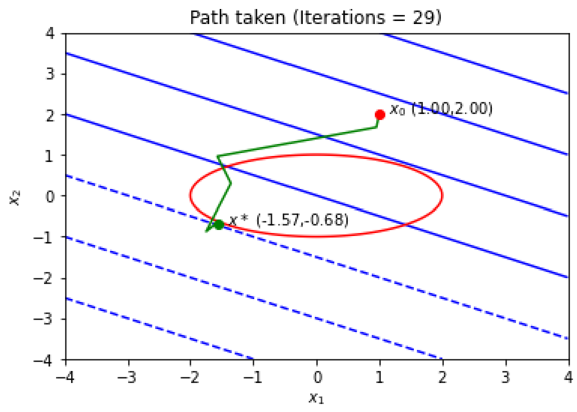
(a) $\mu_b = 1$, $\mu_p = 2$.



(b) $\mu_b = 1$, $\mu_p = 20$.



(c) $\mu_b = 0.2$, $\mu_p = 15$.



(d) $\mu_b = 10$, $\mu_p = 15$.

Figure B.1: Variation of path taken, and number of iterations required by optimizer with change in optimizer parameters

This page was intentionally left blank.

C
Deflation constraint

C.1. KKT proof

In this section, the proof of how the modified problem, after the introduction of deflation constraint shown in subsection 6.2.1 provides a solution (x^*, y^*) that also satisfies the stationary conditions of the original problem is portrayed. The modified problem is given as [23],

$$\begin{aligned} \underset{x \in \mathcal{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c(x) = 0 \\ & m(x; x_1) \leq y \\ & l \leq x \leq u \end{aligned} \tag{C.1}$$

It is to be proved that, x^* is a regular KKT point to the problem given in Equation 6.1 and that $x^* \neq x_1$. Let a new Lagrange multiplier η be introduced in addition to the ones in Equation 6.2 associated with the deflation constraint. Therefore the stationary conditions now will be,

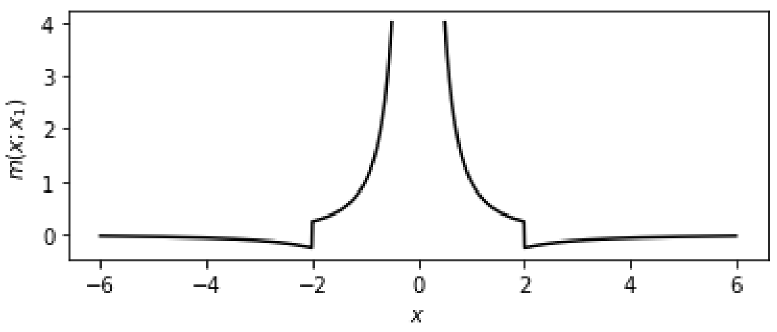
$$\begin{gathered} \nabla_x f(x) + \nabla_x c(x)^T \lambda + z_+ + z_- + \eta \nabla_x m(x; x_1) = 0 \\ \eta = 0 \end{gathered} \tag{C.2}$$

It can be said that η would be 0 at any KKT point, hence the stationary conditions for Equation 6.1 would be satisfied. Since the constraints given in Equation 6.1 are a subset to those in Equation C.1, x^* must be a feasible point in the original problem (Equation 6.1) as well, and the complementary conditions of those constraints must be satisfied.

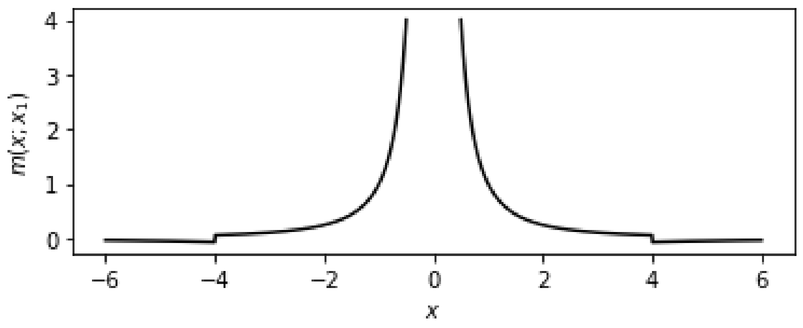
Since the obtained y^* is finite, (x^*, y^*) is feasible to the deflation constraint problem in Equation C.1 and m is bounded from below, then $m(x^*; x_1)$ must be finite, which implies that $(x^* \neq x_1$. This proof can be trivially generalized also to inequality-constrained NLPs, hence the deflation constraint approach is a generic and non-invasive way of utilizing deflation for an optimization problem [23].

C.2. Effect of Parameters

In this section, the plots of the effect of varying the various parameters in Equation 6.11 are displayed. The variation of the parameter σ has already been covered in Figure 6.2.2. Now the variation of the other two parameters r and p are observed.

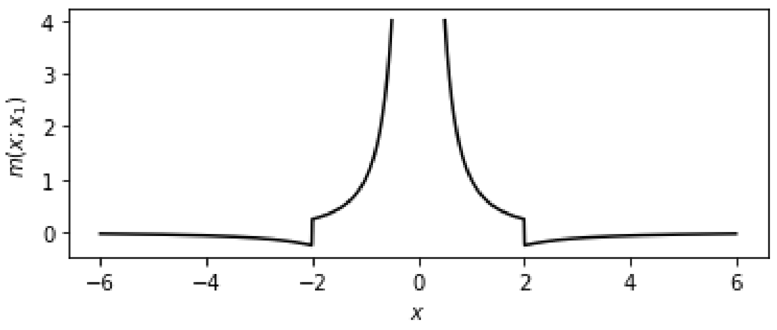


(a) $\sigma = 0$, $r = 2$ and $p = 2$

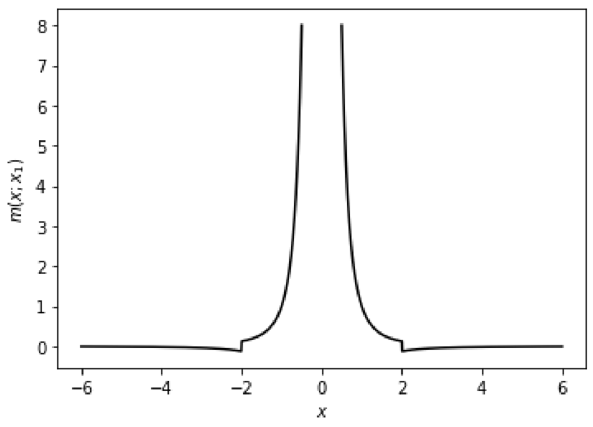


(b) $\sigma = 0$, $r = 4$ and $p = 2$

Figure C.1: The variation in value of deflation constraint represented in Equation 6.11 about a known solution $x_1 = 0$.



(a) $\sigma = 0$, $r = 2$ and $p = 2$

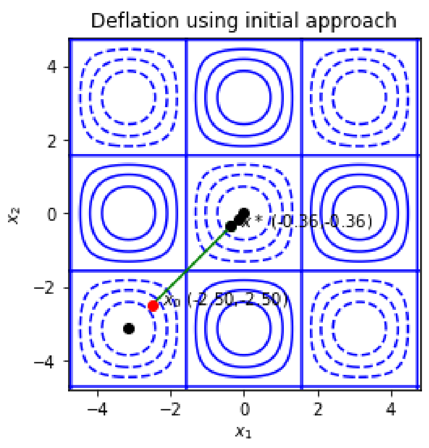


(b) $\sigma = 0$, $r = 2$ and $p = 3$

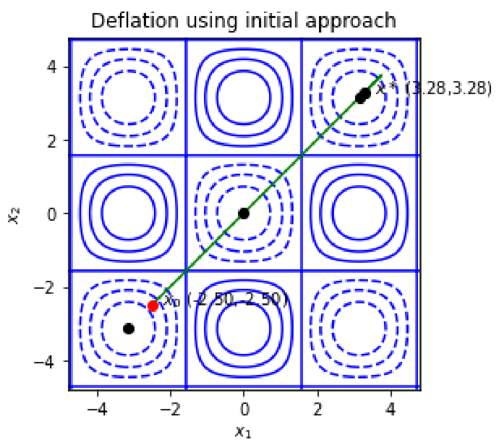
Figure C.2: The variation in value of deflation constraint represented in Equation 6.11 about a known solution $x_1 = 0$.

C.3. Test of deflation constraint

The following images in Figure C.3 show that while using the deflation constraint formulation given in Equation 6.5, how the change in parameters affects the solution(s).



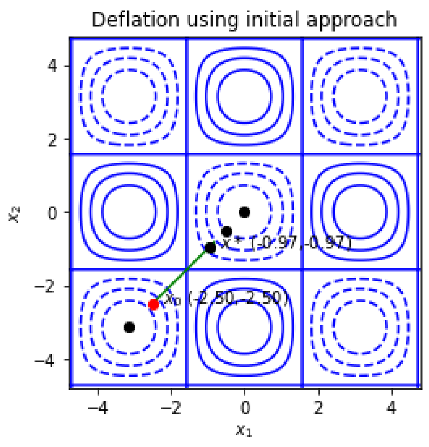
(a) $\sigma = 10$ and $p = 3$



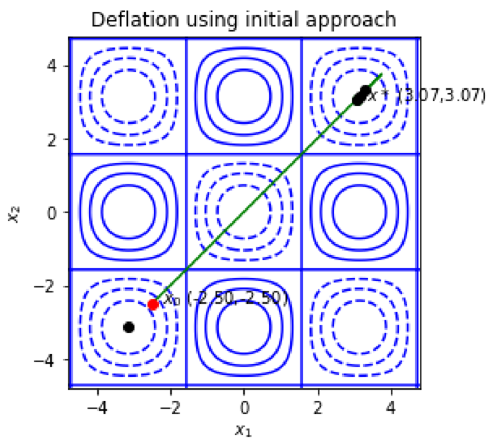
(b) $\sigma = 40$ and $p = 2$

Figure C.3: The solution obtained after deflation of double-cosine function with different deflation parameters using deflation constraint formulation given in Equation 6.5.

The following images in Figure C.3 show that while using the deflation constraint formulation given in Equation 6.7, how the change in parameters affects the solution(s).



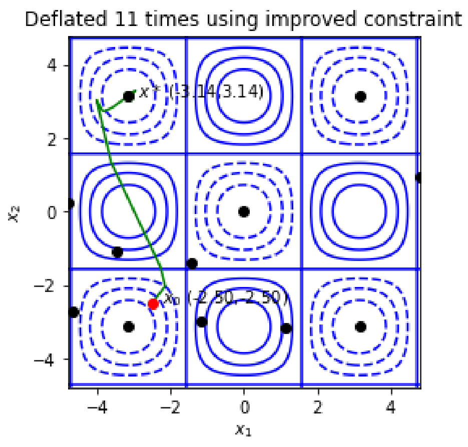
(a) $\sigma = 5$, $M = 6$ and $p = 2$



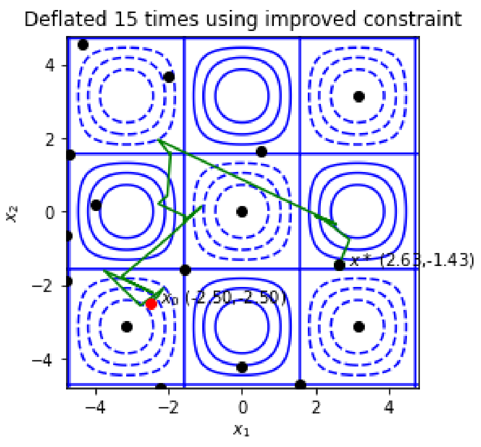
(b) $\sigma = 5$, $M = 20$ and $p = 2$

Figure C.4: The solution obtained after deflation of double-cosine function with different deflation parameters using deflation constraint formulation given in Equation 6.7.

The following images in Figure C.3 show that an increase in the radius of action in Equation 6.9 decreases the number of solutions that needs to be evaluated to find the significant minimas but an excessive radius can cause it to not be able to evaluate a significant minimum.



(a) $\sigma = 0$, $r = 2$ and $p = 2$



(b) $\sigma = 0$, $r = 3$ and $p = 2$

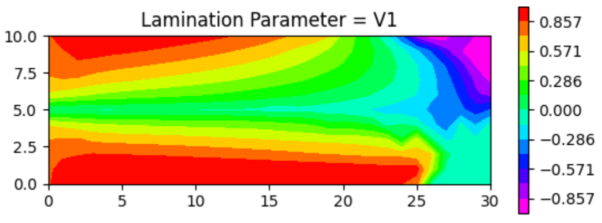
Figure C.5: The solution obtained after deflation of double-cosine function with different deflation parameters using deflation constraint formulation given in Equation 6.9.

In Figure C.3.b, it can be seen that in the top left corner, the vicinity of the two obtained minimas are very close to the significant minima required to be found, and the circles created by both these points engulf the region of the significant minima. Therefore, this would deflate the optimizer away from this region.

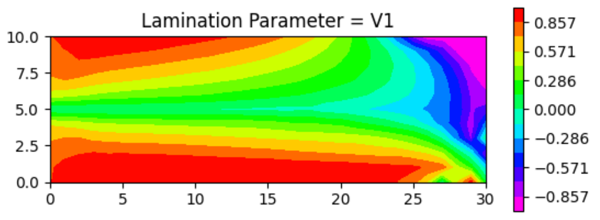
D
Case studies: Additional results and proofs

D.1. Case study 3

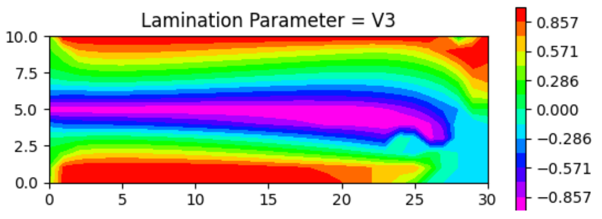
The results for the LP distribution of a balanced VS laminate were compared using two methods, the gradient-based optimizer developed for this thesis, and GA from the pymoo module. This is what the contour plots of the LPs look like:



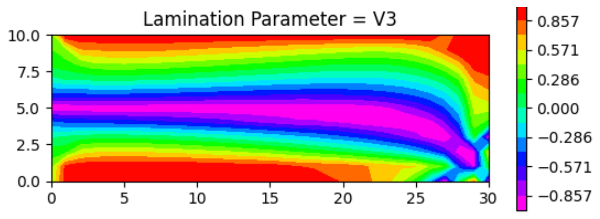
(a) Lamination parameter V_1 obtained with gradient-based optimizer.



(b) Lamination parameter V_1 obtained with GA optimizer.



(c) Lamination parameter V_3 obtained with gradient-based optimizer.

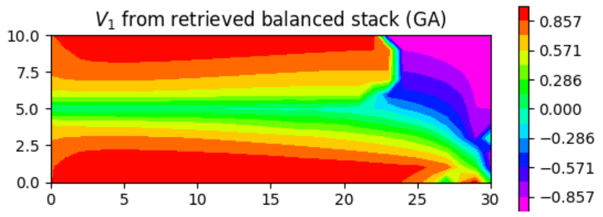


(d) Lamination parameter V_3 obtained with GA optimizer.

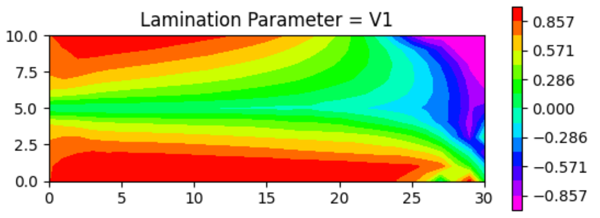
Figure D.1: Distribution of Lamination parameters obtained for balanced Variable stiffness laminate. ($a/b = 3$, and 31×11 nodes)

D.2. Case study 4

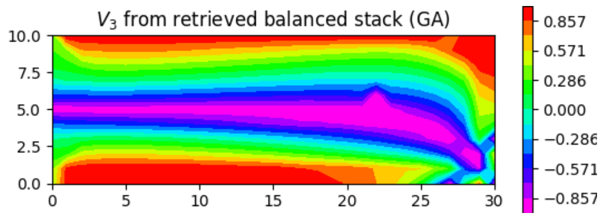
In this section, the results obtained of the LP distribution using the retrieved fiber angles are compared with the Optimum LP distribution as obtained in case study 3 for a balanced laminate. First, the results for the GA are displayed followed by the gradient-based optimizer that utilized deflation to get the ply angles.



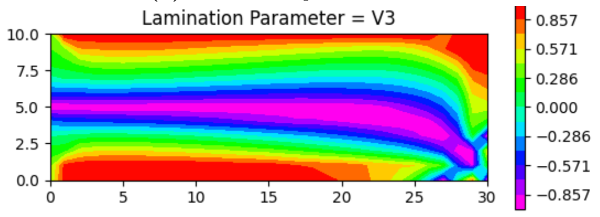
(a) Lamination parameter V_1 obtained with Stack.



(b) Lamination parameter V_1*.

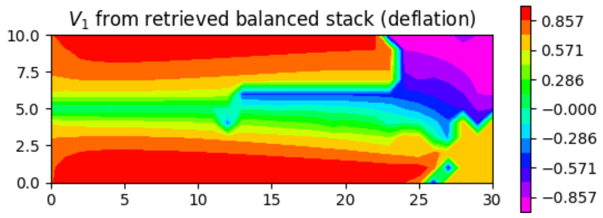


(c) Lamination parameter V_3 obtained with Stack.

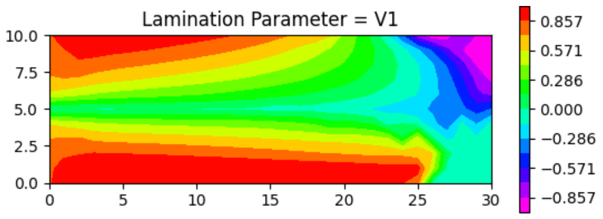


(d) Lamination parameter V_3*.

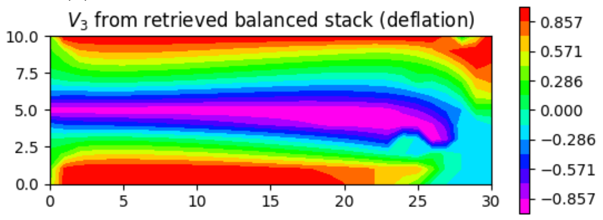
Figure D.2: Distribution of Lamination parameters obtained for balanced Variable stiffness laminate using GA. ($a/b = 3$, and 31×11 nodes)



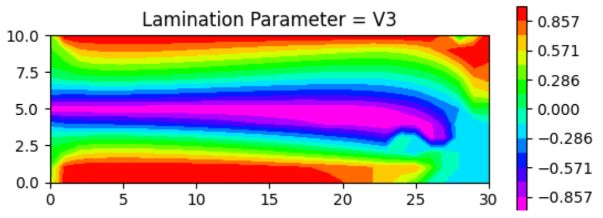
(a) Lamination parameter V_1 obtained with Stack.



(b) Lamination parameter V_1*.



(c) Lamination parameter V_3 obtained with Stack.



(d) Lamination parameter V_3*.

Figure D.3: Distribution of Lamination parameters obtained for balanced Variable stiffness laminate using the gradient-based optimizer. ($a/b = 3$, and 31×11 nodes)