

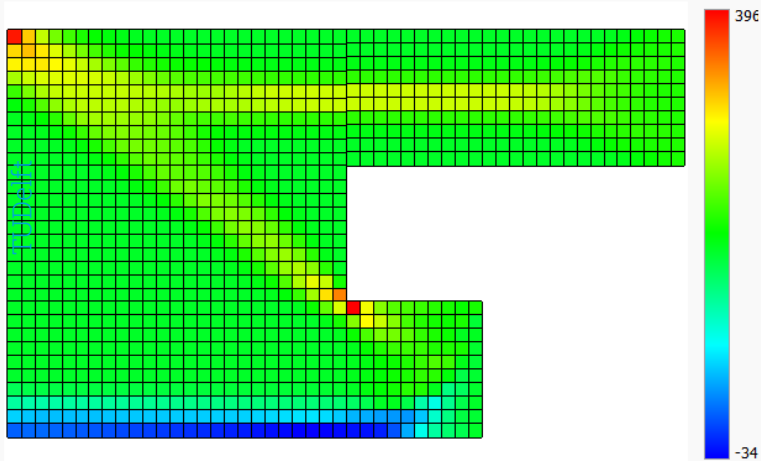
Concurrent Optimization for Laminates with filtered manufacturing constraints

AeroSpace Master Thesis

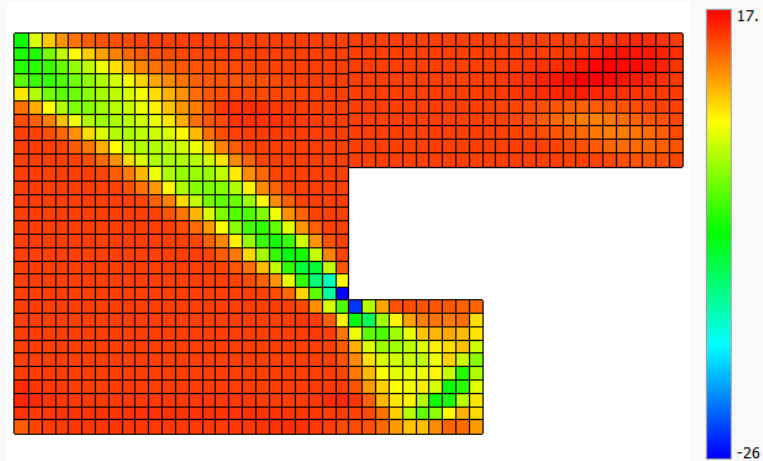
Composite Optimization
Nicolas Fontenai, 5083613

Settings Parameters Run Mesh View Layer View Property View CSV Density Replay Fiber Paths

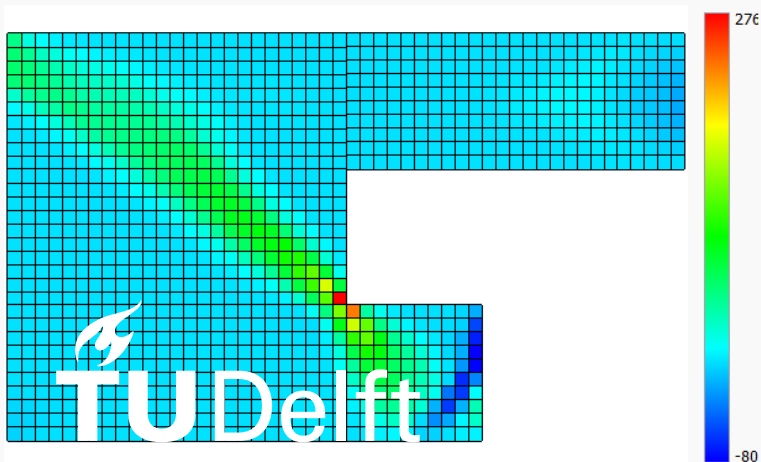
Property Stress XX



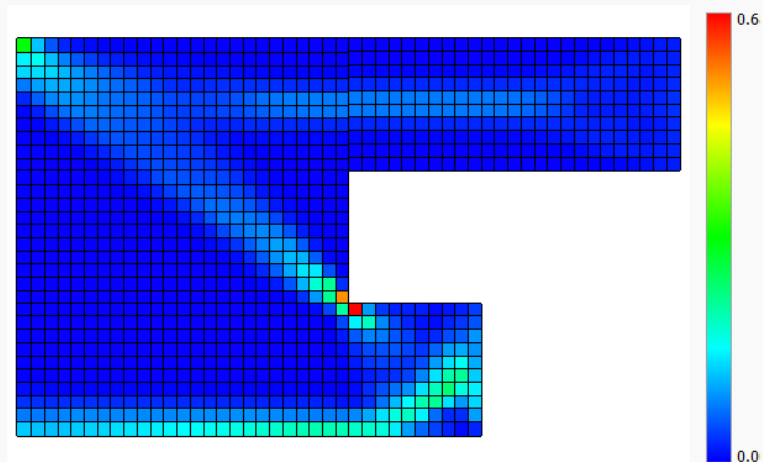
Property Stress XY



Property Stress YY



Property Compliance



Concurrent Optimization for Laminates with filtered manufacturing constraints

AeroSpace Master Thesis

by

Nicolas Fontenai, 5083613

TU Delft Supervisor:

Sergio Turteltaub

SphereCube Supervisor:

Valerio Di Pompeo

Company:

Spherecube

Abstract

The composite additive manufacturing industry keeps growing with every year alongside the need for more light-weight structure problem specific components in EV, wind, space, sports and more industries. This need has come with a great deal of research on the optimization of composites as they offer way more lightweight design possibilities and are more problem directional specific which makes it more efficient. This however comes with some difficulties that are being solved in this thesis namely the manufacturability of it all.

The thesis combines the SIMP method for topology optimization, together with an ANDFO approach to the fiber orientation optimization methods. The SIMP method provides a well defined variable parameter which can be used to retrieve a manufacturable design without gray areas when implemented with a power scale as in this thesis. The ANDFO on the other hand uses normal distribution weight functions to make the fiber also push towards a discrete value without having multiple design parameters per layer.

The model is initially compared for its manufacturability efficiency against other optimization methods like the level-set or heavyside methods, or against the lamination parameters and other methods for the laminate design.

The model, which works with a mix of heuristic and concurrent combination methods, is shown to have verified results tested against other known literature, and has shown to have some amount of increase in its manufacturability and compliance levels. The randomness test however still shows that even with the use of a GCMMA solver the non-convexity of the problem still makes the randomized initialization point have different results however this is mostly viewed for the fiber orientations as they are more susceptible for it.

Contents

Abstract	ii
Nomenclature	xiii
1 Introduction	1
2 Literature Review	3
2.1 Structural Optimization Problem	3
2.1.1 Optimization Approaches	3
2.1.2 Solvers	3
2.1.3 Strategies	5
2.2 Topology Optimization	8
2.2.1 Density Based Method	8
2.2.2 Bi-directional evolutionary structural optimization method	9
2.2.3 Level-set Method	10
2.3 Variable Stiffness Optimization	11
2.3.1 Multi-step Lamination Parameter Approach	11
2.3.2 DMO	12
2.4 Combination Approaches	14
3 Methodology	18
3.1 Finite Element Analysis	18
3.1.1 Classical Laminated Plate Theory	18
3.2 Topology Optimization	23
3.2.1 SIMP	23
3.2.2 Problem	24
3.2.3 Volume Gradient	26
3.3 Fiber Optimization	26
3.3.1 NDFO	26
3.3.2 Problem	27
3.3.3 Adaptive Descent Strategy	28
3.3.4 Curvature Constraint	28
3.4 Solution Strategies	29
3.4.1 Density Filter	29
3.4.2 Fiber Filter	30
3.4.3 Substitution	32
3.5 GCMMA	33
3.5.1 Topology GCMMA	33
3.5.2 VSO GCMMA	34
3.6 Combination Method	35
3.7 Post Processing	36
4 Numerical Implementation	37
4.1 Mesh Generation	37
4.2 Object and Run Setup	41
4.2.1 Boundary Conditions	41
4.2.2 Parameters	42
4.2.3 Properties views	44
4.3 The process	45
4.4 Pre-Processing	45
4.5 Optimization	48

4.6	Post-Processing	56
5	Results, Verification and Discussion	59
5.1	Verification	59
5.1.1	Topology Optimizer	59
5.1.2	Variable Stiffness Optimizer	62
5.1.3	Combined Approach	64
5.2	Analysis of the Parameters	67
5.2.1	Topology filter radius	67
5.2.2	Fiber Orientation filter radius	69
5.2.3	Fiber Orientation filter power	72
5.2.4	Combined effects	73
5.3	Random Initialization points	79
5.4	CBeam	81
6	Conclusion and Recommendations	92
6.1	Recommendations	93

List of Tables

4.1	Coordinates of mesh points	38
4.2	Mesh Lines	39
4.3	Mesh Surfaces	40
4.4	Model parameters and descriptions	42
4.5	Selectable property views and descriptions	44
4.6	Calculate Candidate constitutive matrix	45
4.7	Calculate Constitutive matrices of candidate angles	46
4.8	Calculate Element Centroids	46
4.9	Calculate Element Areas	46
4.10	Find Element Neighbors	47
4.11	Calculate Element Neighbors weights	47
4.12	Precompute derivative shape functions	47
4.13	Precompute Jacobians and related matrices	48
4.14	Precompute Jacobians and related matrices	49
4.15	Filter of Density Design Variable	50
4.16	Filter of Fiber Design Variable	50
4.17	Constitutive matrix calculation of the fiber orientations	50
4.18	SIMP Method application on the constitutive matrices	51
4.19	Calculation of the ABD matrices from the constitutive matrices	51
4.20	Calculation of the element's stiffness matrix	52
4.21	Calculation of the global stiffness matrix	52
4.22	Calculation of the global stiffness matrix	52
4.23	Calculation of the strain matrices	53
4.24	Calculation of the stresses matrices	53
4.25	Calculation of the strain energy matrices	54
4.26	Calculation of the global compliance matrix	54
4.27	Calculation of the density gradient matrix	54
4.28	Calculation of the density filtered gradient matrix	55
4.29	Calculation of the fiber orientation gradient matrix	55
4.30	Calculation of the fiber orientation gradient matrix with filtered orientation	56
4.31	Calculation of the fiber orientation gradient matrix	56
4.32	Generate streamline function from domain and fiber orientations	58
5.1	specification of rectangular plate problem	59
5.2	specification of rectangular plate problem	61
5.3	specification of MBB beam problem	62
5.4	specification of MBB beam problem	63
5.5	specification of MBB beam problem	65
5.6	specification of MBB beam problem	66
5.7	specification of rectangular plate problem	67
5.8	Compliance change on filter radius	69
5.9	specification of rectangular plate problem	69
5.10	Compliance change on filter radius	71
5.11	Compliance change on filter radius	73
5.12	Compliance change on filter radius	75
5.13	Compliance change on filter radius	77
5.14	Compliance change on filter radius	78
5.15	specification of rectangular plate problem	79
5.16	Compliance change on filter radius	81

5.17 CBeam Material Properties	81
5.18 specification of rectangular plate problem	82
5.19 Compliance change on element size	84
5.20 Compliance change on volume fraction	86
5.21 Compliance change on curvature constraint	88
5.22 Compliance change rate on penalty parameters	89
5.23 Comparative compliance analysis	91

List of Figures

2.1	Linear Filter Strategy	6
2.2	nonlinear Filter Strategy	7
2.3	SIMP (Solid Isotropic Material with Penalization) Method	8
2.4	BESO Method example Applied by Xiaolei Yan et al.	9
2.5	Movement of structural Boundary in the Level-set Method	10
2.6	The Lamination Parameter method used by Eralp Demir et al. [8]	11
2.7	Graph of the weighing coefficients ω_{ij, P^c} corresponding to different P^σ	13
2.8	The 3 stages for the deviation penalty parameter P^σ during optimization in ANDFO	13
2.9	Multi-Step Methods using Lamination Parameters	14
2.10	Example of a Hierarchical approach to the optimization used by J.H.S. Almeida et al. [2]	15
2.11	Example of a concurrent approach used by Zuryi Duan et al. [10]	16
2.12	Example of the level-set approach with path optimization used by T.Xu et al. [33]	17
3.1	quad element representation in FEA	20
3.2	Gauss Integration Points in a quad element for FEA	21
3.3	The 3 decent stages for the deviation penalty parameter	28
3.4	Density Filter on Topology Optimization	29
3.5	nonlinear Filter Strategy	31
3.6	Combination Algorithm for the Laminate Optimization	35
4.1	Quad element representation in FEA	37
4.2	Definition of the Objects Design Space before Mesh Generation	38
4.3	Definition division in design space before mesh generation	38
4.4	Location of the points set for mesh generation	39
4.5	Line direction setup for the mesh generation	40
4.6	The mesh result from the example	41
4.7	Boundary Conditions UI	41
4.8	Parameter page of the program	43
4.9	Runpage with property selection to view	44
4.10	Control Panel of the Simulation Run	49
4.11	Fiber Path UI Page	57
5.1	Beam problem used by Sigmund Ole [24]	60
5.2	Beam problem result	60
5.3	Beam problem used by S. Bulman et al.[7]	61
5.4	Beam problem results from 2 nd test	61
5.5	1 st test of beam problem for VSO verification	62
5.6	Result from paper by Yi Lui et al.[17]	63
5.7	Results for 1 st test of VSO verification	63
5.8	2 nd test for VSO verification used by C.Y. Kiyono et al.[15]	63
5.9	Result from paper by C.Y. Kiyono et al.[15]	64
5.10	Results for 2 nd test of VSO verification	64
5.11	1 st test for combined optimization verification	64
5.12	Result from paper by Zunyi Duan et al.[10]	65
5.13	Result for 1 st test off combined optimization verification	65
5.14	2 nd test for the combined optimization verification used by Furong Zie et al.[32]	66
5.15	Results for 2 nd test of the combined optimization verification	66
5.16	Problem setup for parameter analysis	67
5.17	The parametrization of the density filter radius	68

5.18 Fiber orientation filter radius $R^\theta = 1.0$	70
5.19 Fiber orientation filter radius $R^\theta = 2.0$	70
5.20 Fiber orientation filter radius $R^\theta = 4.0$	70
5.21 Fiber orientation filter radius $R^\theta = 5.0$	70
5.22 Fiber orientation filter radius $R^\theta = 10.0$	71
5.23 Checkerboard pattern for the fiber orientation optimizer	71
5.24 Fiber orientation filter power $p^\theta = 1.0$	72
5.25 Fiber orientation filter power $p^\theta = 2.0$	72
5.26 Fiber orientation filter power $p^\theta = 3.0$	72
5.27 Fiber orientation filter power $p^\theta = 5.0$	73
5.28 Combined approach with density filter $R^\rho = 1.0$	74
5.29 Combined approach with density filter $R^\rho = 2.0$	74
5.30 Combined approach with density filter $R^\rho = 3.0$	74
5.31 Combined approach with density filter $R^\rho = 5.0$	74
5.32 Combined approach with fiber orientation filter $R^\theta = 1.0$	75
5.33 Combined approach with fiber orientation filter $R^\theta = 2.0$	75
5.34 Combined approach with fiber orientation filter $R^\theta = 3.0$	76
5.35 Combined approach with fiber orientation filter $R^\theta = 5.0$	76
5.36 Combined approach with fiber orientation filter $R^\theta = 10.0$	76
5.37 Combined approach with fiber orientation filter power $p^\theta = 1.0$	77
5.38 Combined approach with fiber orientation filter power $p^\theta = 2.0$	77
5.39 Combined approach with fiber orientation filter power $p^\theta = 3.0$	78
5.40 Combined approach with fiber orientation filter power $p^\theta = 4.0$	78
5.41 Combined approach with fiber orientation filter power $p^\theta = 5.0$	78
5.42 Random initialization test for $\mu_1 = 0.00001$	80
5.43 Random initialization test for $\mu_1 = 0.0001$	80
5.44 Random initialization test for $\mu_1 = 0.001$	80
5.45 Random initialization test for $\mu_1 = 0.01$	80
5.46 CBeam Problem Definition	81
5.47 CBeam Mesh Generations	82
5.48 Mesh generation with elementsize 4mm	83
5.49 Mesh generation with elementsize 2mm	83
5.50 Mesh generation with elementsize 1,25mm	83
5.51 Mesh generation with elementsize 1mm	84
5.52 Volume Fraction of 0.3	85
5.53 Volume Fraction of 0.4	85
5.54 Volume Fraction of 0.5	85
5.55 Volume Fraction of 0.6	86
5.56 Curvature Constraint of 2mm	87
5.57 Curvature Constraint of 4mm	87
5.58 Curvature Constraint of 5mm	87
5.59 Curvature Constraint of 8mm	88
5.60 Curvature Constraint of 10mm	88
5.61 Topology Optimization	90
5.62 Fiber Optimization	90
5.63 Combined Optimization	90

Nomenclature

Abbreviation

FRC	Fiber Reinforced Composites	MMA	Method of Moving Asymptotes
SIMP	Solid Isotropic Material with Penalization	GCMMA	Globally Convergent Method of Moving Asymptotes
BESO	Bi-directional Evolutionary Structural Optimization	NDFO	Normal Distribution Fiber Optimization
DMO	Discrete Material Optimization	ANDFO	Adaptive Normal Distribution Fiber Optimization
EAOT	Equivalent Anisotropy Orientation Tensor	CLPT	Classical Laminate Plate Theory
FEA	Finite Element Analysis	BC	Boundary Condition
CSR	Compressed Sparse Row	KKT	Karush-Kuhn-Tucker
VSO	Variable Stiffness Optimization	DOF	Degrees of Freedom

Symbol	Description	Symbol	Description
--------	-------------	--------	-------------

Design Variables

ρ	Density design variable	θ	Fiber orientation design variable
$\tilde{\rho}$	Filtered density variable	$\tilde{\theta}$	Filtered fiber orientation variable
x	General design variable		

Matrices and Vectors

$\mathbf{K}$	Global stiffness matrix	$\mathbf{K}_e$	Element stiffness matrix
$\mathbf{A}$	Extensional stiffness matrix	$\mathbf{B}$	Coupling stiffness matrix
$\mathbf{D}$	Bending stiffness matrix	$\mathbf{Q}$	Reduced stiffness matrix
$\bar{\mathbf{Q}}$	Transformed stiffness matrix	$\mathbf{B}_e$	Strain-displacement matrix
$\mathbf{u}$	Displacement vector	$\mathbf{F}$	Force vector
$\mathbf{N}$	In-plane force resultants	$\mathbf{M}$	Bending moment resultants
$\mathbf{J}$	Jacobian matrix		

Material Properties

E_1, E_2	Young's moduli	G_{12}	Shear modulus
ν_{12}, ν_{21}	Poisson's ratios	t	Layer thickness

Physical Quantities

C	Compliance	σ	Stress
ε	Strain	τ	Shear stress

γ	Shear strain	κ	Curvature
V	Volume	z	Thickness coordinate

Optimization Parameters

p	Penalty power (SIMP)	P^σ	Normal distribution deviation
P^c	Candidate fiber angle	N_{cand}	Number of candidate angles
R_ρ	Density filter radius	R_θ	Fiber filter radius
ω	Weight coefficient	f_V	Volume fraction
f	Objective function	g	Constraint function
U, L	Upper/lower asymptotes		

Indices and Counters

e	Element index	l	Layer index
N_{elem}	Number of elements	N_{layers}	Number of layers
k	Iteration index	i, j	General indices

1

Introduction

In 2026, more than ever, the automotive and energy sectors are increasing their use of AFP (automated fiber placement) composites to build more lightweight EVs, as every weight saved in an EV chassis reduces the necessity for large batteries and in turn reduces vehicle costs [20]. Also in the energy sector, weight matters as better TFP (tailored fiber placement) in wind turbines can decrease its necessary weight and thus increase its efficiency. Here, composite optimization allows us to place carbon fiber along primary load paths [25].

Currently, the literature shows much progress in composite optimization with many different methods solving this non-convex problem. Many methods in the literature neglect manufacturability and focus on the theoretical optimizable result which cannot be used practically in the real world. Some studies try to add some manufacturability through filtering of design variables, but the problem remains non-convex and often provides theoretical specific results.

This thesis tackles and develops existing methods that have proven to have improved manufacturability and optimization. It bridges the gap where some optimizers like ANDFO optimizer lack in their fixed curvature constraints and amplifies the need for a more robust model that can produce results specific for certain machines with their curvatures constraint. It looks at the method more in depth and discusses its weaknesses and strength as well as provides some different methods to solve certain weaknesses. The thesis initially goes over some of the current available and most promising methods as well as shows an in depth look into optimization itself, so the reader has a good comprehension as to why this problem is so complex to tackle.

The literature study focuses on general optimization techniques and provides the reader with a good understanding as to why a certain method is better for this problem than another. It provides a study on the different solvers used and the strategies applied that can help maximize the manufacturability. It continues with a section on different topology optimization methods that have been proven to have great result. On top of the Topology optimization it also covers the VSO (Variable stiffness optimization) methods and a section on different methods for combining both the topology and the VSO together in 1 optimizer.

In the Methodology section the most manufacturability fitted methods from the literature study are explained in depth and used to build the optimizer used by this thesis. It explains the laminate theory that is used in the mathematics, an in depth look into the math behind the topology optimization with a problem definition and sensitivity analysis. Similar to the topology optimization a mathematical definition and analysis on the VSO is done. To face the manufacturability, some strategies are explained to achieve these manufacturable results for our methods. Finally, the solver chosen is implemented with the optimizing methods chosen and then the methods are all combined into an algorithm to achieve the best results possible.

After the methodology section an in depth look at the program and numerical implementation of the methodology is applied to understand and reproduce the code if wanted. The thesis then ends with

a section where the optimizer is tested and an analysis on the different parameters for manufacturability are looked at in depth to provide the most optimal results from the program, after which different examples and interesting tests are applied to discuss the programs capabilities.

2

Literature Review

2.1. Structural Optimization Problem

2.1.1. Optimization Approaches

Many optimization approaches have been used by mathematicians, engineers and other professions to obtain optimal results to desired systems. Many of these have their own strengths and weaknesses in how well they can be used by a system, how fast they provide the optimal result, or how highly constraint the problem can be.

Some reviewed approaches, provided later on, are used in FRC (Fiber Reinforced Composites) optimizations to obtain optimal composite results with high structural performances [35]. They target key properties that represent their objectives by simplifying them into mathematical models that optimization systems can use. The objective functions are mathematically represented by f , and should be minimized or maximized in respect to the set design variables x . This optimization is represented as such:

$$\begin{aligned} \max f(x), \\ \min f(x), \end{aligned} \tag{2.1}$$

In most optimizations, constraints are required to be meaningful. They set limitations on the design variables to achieve more goal specific results. Mathematically they are represented by an equality or inequality [21]:

$$\begin{aligned} g(x) &\geq 0, \\ h(x) &= 0, \end{aligned} \tag{2.2}$$

In practical optimization methods, these constraints set the rules for the system. For example, in structural or topology optimization, constraints often control the total material used, enforce minimum feature sizes, or ensure mechanical feasibility. The constraint functions and the objective functions are then used together in a *Solver*, which uses these functions and different methods to find the optimal solution for them.

2.1.2. Solvers

The different optimization problem have different solving approaches to result in their respective converged goals. For convex problems very efficient optimizer are the gradient based optimizers.

Gradient descent optimization

Convex linear equations can iteratively, and efficiently, be solved using a gradient descent optimization method.

The method uses the tangential slope or gradient of the current design to evaluate in which direction the optima are located. The gradient is iteratively calculated from the sensitivities of the design variables with respect to their objective function. The slope is then followed with a certain user-set step for the next iteration. After which, the loop starts again and the sensitivities and gradients are recalculated until convergence.

$$x^{k+1} = x^k - \text{step} \nabla f(x^k) \quad (2.3)$$

In the unconstrained equation for the gradient descent method, the x^{k+1} represent the next iteration of the design variable x^k , step the step size at which the gradient $\nabla f(x^k)$ is applied.

The method is computationally very fast in finding its optima, but has some problems as it can fall into local optima when the problem is non-convex or non-linear. The MMA (Method of Moving Asymptotes) method improves on the gradient decent method with an adaptive step control and good handling of the constraints

MMA

The non-convexity of the problem stays also in MMA solver. However, MMA uses known convex functions to approximate the current evaluation of the system at an iteration and thus provides for every operation, a singular solvable solution to work toward, as shown by Xueyan Hu et al. [12].

The method uses asymptotes to limit the change in the design variable at every iteration. These asymptotes are used to create the approximation function around the current value of the design variable:

$$\begin{aligned} \tilde{f}^k(x) &= \sum_{j=1}^n \left(\frac{p_j^k}{U_j^k - x_j} + \frac{q_j^k}{x_j - L_j^k} \right), \\ \tilde{g}_i^k(x) &= \sum_{j=1}^n \left(\frac{r_{ij}^k}{U_j^k - x_j} + \frac{s_{ij}^k}{x_j - L_j^k} \right) \end{aligned} \quad (2.4)$$

The $\tilde{f}^k(x)$ is the convex approximation, of the objective function, and $\tilde{g}^k(x)$, of the constraint function, used in MMA to solve for the next iteration. U_j^k and L_j^k being respectively the upper and lower asymptotes, and p_j^k and q_j^k are two variables set to fulfill the gradient equation:

$$\frac{d\tilde{f}}{dx_j} \approx \frac{df}{dx_j} \quad (2.5)$$

This approximation is simultaneously done for the constraint functions of the system after which the approximated system can be solved analytically resulting in 1 solution for the minima. A certain step towards this solution is then followed, and the asymptotes are grown or shrank based on if the design updates need to be more locked in or if they have the room for bigger steps.

The MMA method has better control of its step-size and can lock into minima faster than the gradient descent method. It can also more efficiently use the constraints in its analysis. However the MMA method still considers the gradient as a compass to follow blindly toward the optimum, which can cause divergences or neglect constraints. GCMMA (Globally Convergent Method of Moving Asymptotes), on the other hand, uses the same system as MMA but with more doubt and mathematical restrictiveness towards following the gradient.

GCMMA

The GCMMA solver adds a conservative addition to the constraints where MMA might have a lower approximation. It's a method highly used in composite optimization where a lot of constraints like stress and buckling need to be taken into account [12].

The application goal of GCMMA is to have the approximation function of the constraint to be higher than the actual function within the feasible domain.

$$\tilde{g}_i(x) \geq g_i(x) \quad (2.6)$$

This state is achieved using an additional term γ_j^k which increases the approximation function so that the equation in 2.6 holds true.

$$\tilde{g}_i^k(x) = \sum_{j=1}^n \left(\frac{r_{ij}^k}{U_j^k - x_j} + \frac{s_{ij}^k}{x_j - L_j^k} \right) + \gamma_j^k \quad (2.7)$$

To update the asymptotes the method also uses an adaptive asymptote update system, instead of a constant step, to ensure the asymptotes don't take too big steps when not necessary.

Others

Next to gradient based, deterministic, solvers there are also solvers using direct search methods, where the function as a global is evaluated and they use patterns to explore the design space. This method can for example help with finding more the global optimum in the end as it is not stuck in a certain direction of the system at the initial pattern compared to gradient based.

Evolutionary algorithms use population-based optimization, as used by ZhenZhou Wang et al.[30], to fill the design space with many optional re-combinations and evaluates them with swarm like intelligence to retrieve the optimal results. e.g. it analyzes a set of different candidate solutions randomly selected and uses the one with the best objective value to evolve into the next iteration of candidate solutions.

The choice of solver shows the blueprint into solving the system's optimization. However, many results that come out of these solvers are not useful as the solver only considers the optimal for the objective and the constraints. To solve this certain *strategies* can be implemented to provide feedback to the solver of which type of results are not desired.

2.1.3. Strategies

Because of the simplified nature of optimizers, they can often result in non-manufacturable or reality inconsistent optima, which shows the complexity in which reality based manufacturable optimizers lie. To tackle these problems, strategies have been implemented to either push design variables towards certain results or filter them to ensure meaningful results. The penalization is often used for topology or other optimizers

Penalization

Some optimizers implement the design variables using a continuous domain, to have an easier analytical way to efficiently arrive at a result. However some of these design variables have as a result non-physical solutions.

Penalization uses mathematical methods to push these design variables towards more physically meaningful values. Power penalization strategy x^p uses a power p on the design variable to push the value towards a 0, 1 solution as used by Bendsoe et al. [6]. Another strategy implemented to push values towards 0 nonexistent and 1 existent value is the heavy side penalization.

$$\tilde{x} = \frac{\tanh(\beta\eta) + \tanh(\beta(x - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1 - \eta))} \quad (2.8)$$

In this penalization the β parameter determines the steepness of the Heaviside function similar to the power value. Weighting coefficients have also been used as penalization strategies where certain values in the domain have weights to them and these weight which are continuous between 0 and 1 are then penalized via methods used above.

Even with penalization, gradient on design variables can vary a lot for different elements in the mesh they have some independence towards each other. Filtering strategies can help make neighboring element in a mesh dependent on each other.

Filtering

When two neighboring elements have different signs of gradients it can sometimes cause a high shift in value from one element to its neighbor.

Filtering strategies use mathematical functions to incorporate neighboring values into its final next iteration value.

Two types of filtering strategies implemented in optimizations are the design variable filtering, and the sensitivity filtering. They use the average of the weighted neighbors to set the average value of the density variable or sensitivity:

$$\begin{aligned}\tilde{x}_i &= \frac{\sum_{j \in N(i)} \omega_{ij} x_j}{\sum_{j \in N(i)} \omega_{ij}}, \\ &\text{or} \\ \frac{\tilde{df}}{dx_i} &= \frac{\sum_{j \in N(i)} \omega_{ij} \frac{df}{dx_i}}{\sum_{j \in N(i)} \omega_{ij}},\end{aligned}\tag{2.9}$$

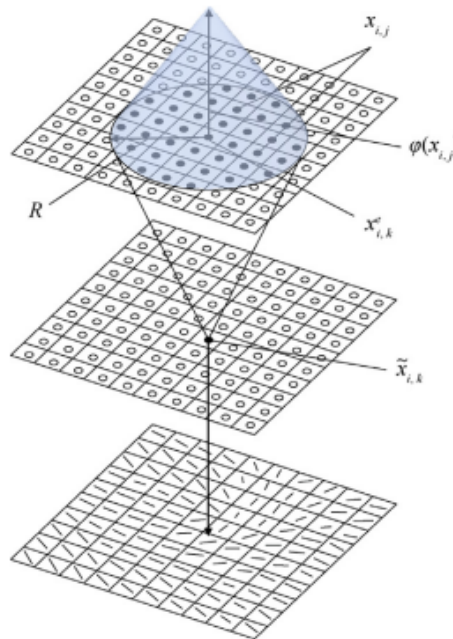


Figure 2.1: Linear Filter Strategy

The design variable filter helps with removing checkerboard patterns or to make sure that surrounding neighbors have similar dependent values for the variable. The sensitivity filter helps by ensuring that neighboring elements adapt gradually together to the system and thus improving convergence to a feasible solution.

There are also filtering strategies that use power penalties on the weights to increase the effectiveness of close by neighbors and decrease the effectiveness of further away neighbors.

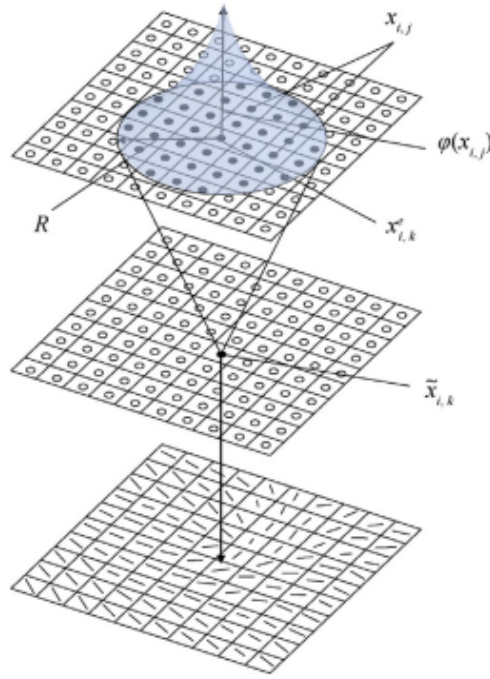


Figure 2.2: nonlinear Filter Strategy

Adaptivity

Adaptivity refers to the broad selection of strategies that deal with changing variables or parameters in adaptation to the current system.

Parameter adaptability, where parameter e.g. power penalty or gradually increased over the iterations so that the parameter initially leaves room for switching before locking down value into their desired values.

$$p_{k+1} = p_k + \Delta p \quad (2.10)$$

This strategy is widely used to leave room for parameters to initially explore and then get fixed to their respective values. It is also used by uniquely changing parameters for certain design variables tailored to the situation of those specific design variables for example to converge more to a solution in one element and to remove oscillation in another at the same time.

Multiscale decomposition

It is difficult to put many simplifications of reality into optimizers without losing certain important information. Multiscale decomposition of optimizers can help simplify the problem into multiple steps where each step results in a more physically feasible realistic representation of the result.

This strategy helps to reduce the complexity of the problem and aids to improve the convergence of the result. The typical steps in which the optimizers are set in is initially in macro-scale where the form of the design is obtained. Secondly, microscale how do optimize what each element now is in itself for the problem. Finally, some step to refine the solution and retrieve physically doable results.

In general many different strategies are used for a wide variety of optimizers in different categories like economics, engineering or others. On this thesis the focus lies on finding optimal composites which is split in 2 main types of optimizers a topology optimizer and variable stiffness optimizer. First up, the topology optimizer approaches are reviewed in depth.

2.2. Topology Optimization

The initial part of any structural optimization, is to know where material has to be placed for the structure to hold up to certain stresses and loads, like designing a bridge out of Legos. This often comes with a constraint on the amount of material the system can use, as to efficiently use the available resources instead of just filling everything up.

A simple way to mathematically have a continuous value of whether or not material should exist on a certain location is by implementing a density method, where the density represents the amount of material used in that location.

2.2.1. Density Based Method

The density based method sets the density variable onto the material properties of the system. To simplify, the design variable is clipped between 0 and 1 and represents the amount at which the material properties of an element are present, 1 being fully present 0 representing no material properties in that element.

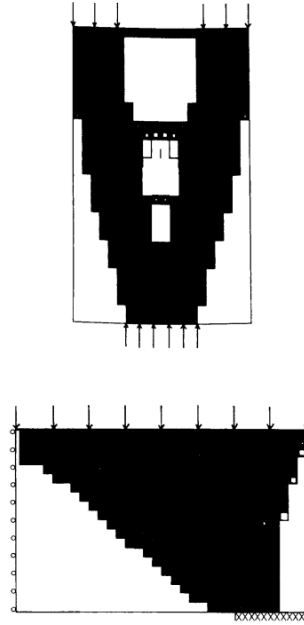


Figure 2.3: SIMP (Solid Isotropic Material with Penalization) Method

The most used case for this is the SIMP Method, introduced by Bendsoe and Sigmund [5]. It uses a density variable ρ penalized with a power parameter p to guide the variable solution towards a binary solution (0 or 1).

$$E = E_{min} + \rho^p (E_0 - E_{min}) \quad (2.11)$$

The minimum value E_{min} is used to avoid singularities when reaching 0.

In FRCs, this method extends to the fiber orientation θ that defines the direction of the material properties inside the element. This is done by assembling the global stiffness matrix K with each element's stiffness K_e matrix subjected to the variable density ρ , as shown by Delin J. et al. [13].

$$K(\rho, \theta) = \sum_{e=1}^N (\rho_e)^p K_e(\theta_e) \quad (2.12)$$

Where e represents the element, and N the number of elements. This Method is simple and keeps the number of the design variables of the density limited to 1 for element. However this does limit the optimizer by making the existence of material in all layers of the element uniform.

A different setup for the SIMP method, represented by Zheng Qiu et al.[23], distinguishes the density variable for each layer in the element. This sets up a system for which the layup itself can be optimized for as well. This is mathematically achieved by applying the density variable on the elasticity matrix D of the layer inside the element.

$$D_{l,e}^{mac} = (\rho_{l,e}^{mac})^p \times D_{l,e}^{mic} \quad (2.13)$$

Where l and e represent the layer and element respectively, ρ the density variable, and p the penalty power. While this provides a more specific construction of the composite with the optimizer it also results in a more heavy computational cost, as the design variable is multiplied by the amount layers.

Even with the penalty power there are still some gray regions that come as results out of SIMP methods and that is where the BESO (Bi-directional Evolutionary Structural Optimization) method is better in.

2.2.2. Bi-directional evolutionary structural optimization method

Instead of continuous density used by the SIMP method, the BESO method uses a discrete design variable x which can either represent 1 (solid) or 0,001 (void).

The method optimizes the design by setting each element to one of the discrete values after each iteration. This is determined by how much an element is contributing to the objective function. After this a cutoff is done based on the resource constraint provided, which results in a certain top performing percentage of elements gets turned into a solid and the other into a void. This is then done iteratively recalculating the performances of each element after the cutoff sequence [26].

In a paper by Xiaolei Yan et al. [34], they used the BESO Method, to optimize the topology of a structure with 3 design variables, the macrostructure, microstructure and orientation of microstructure. The objective function was the natural frequency of the system:

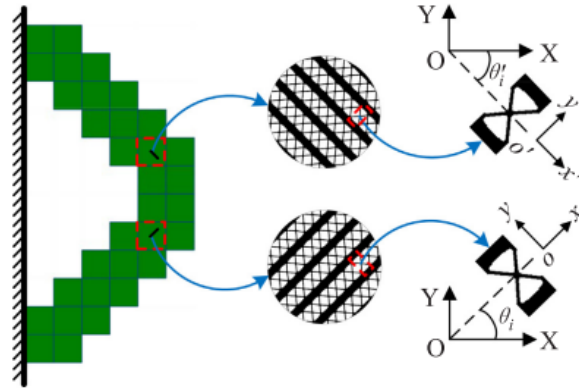


Figure 2.4: BESO Method example Applied by Xiaolei Yan et al.

$$\begin{aligned} & \max(\omega_k) \\ & [K(x, y) - \omega_k^2 M(x, y)] u_k = 0 \\ & x, y = 1 \text{ or } 0.001 \\ & -\frac{\pi}{2} < \theta \leq \frac{\pi}{2} \end{aligned} \quad (2.14)$$

where x and y the macroscale and microscale structure design variables are, ω_k is the natural frequency, K and M are the macro structural stiffness and mass matrices, u_k is the displacement matrix, and θ_k is the orientation of the microstructure.

The BESO method is a strong method for direct representations for either a void or solid, it results in no gray areas and a clear distinction. But even with the BESO method, there are still some boundary difficulties like irregularities or blocky-ness. The level-set method on the other hand optimizes its shape using smooth boundaries

2.2.3. Level-set Method

While the SIMP and BESO Method implement their optimization using a meshed system, the level-set method uses mathematical surface boundaries equations in attempt to receive a less jagged and a more smooth boundary result [29].

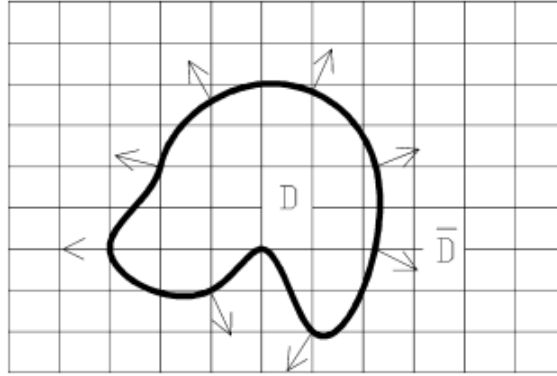


Figure 2.5: Movement of structural Boundary in the Level-set Method

The method uses a boundary function $\phi(x)$, also called the level-set function, that defines the structure of the object by setting:

$$\begin{aligned}\phi &> 0, \text{ solid} \\ \phi &< 0, \text{ void} \\ \phi &= 0, \text{ boundary}\end{aligned}\tag{2.15}$$

In each iteration, the system evaluates its sensitivities on the boundary resulting in the boundary velocity field V_n . This is then used with the gradient of the level-set function in the Hamilton-Jacobi equation to update the next iteration of the shape function as follows:

$$\frac{d\phi}{dt} + V_n |\nabla \phi| = 0\tag{2.16}$$

In this equation, $\frac{d\phi}{dt}$ is the change of the shape functions over the iterations.

After the initial shape is set, the function doesn't allow any more protrusion with this method as only the boundary can shift and no void can form inside. This is why this method often gets initialized with a porous shape where the pores are already inside the boundary and can be used to grow different structures with holes.

This method has been proved to have smoother boundaries and higher sharpness in edges compared to the SIMP and BESO methods. However, the method has a high computational cost due to its complex mathematical implementation, and while you can add initial pores, once a boundary is formed no more holes can form inside it.

The many approaches in topology optimizers have a wide variety of strategies to get that optimal macroscale for the structure set. A more complex system is the microstructure or the variable stiffness optimizer which deals with more design variables and higher non-linearity.

2.3. Variable Stiffness Optimization

In FRCs there are also optimizations inside the composite anisotropic properties and layups. These help improve material structures more specifically for their respective load demands. They help the user not only optimize the topology but also their stacking sequence and fiber placement for optimal structural performance.

So where topology optimization optimizes the placement of material itself, Variable stiffness optimization optimizes the variations in stiffness properties throughout the object due to fiber angles and thus anisotropic material.

There are a lot of studies on different approaches to optimize these fiber paths. They use the ABD matrix of a laminate inside an element to retrieve its stiffness properties from the layer stacks or parameters. The ABD-Matrix provides the constitutive properties of the laminate from where the stiffness matrix is assembled.

This simplifies the VSO optimization there are multistep approaches where the system gets simplified in each step to end up with the fiber paths in the end a blueprint of steps.

2.3.1. Multi-step Lamination Parameter Approach

Composites consist of 1 or more layers of lamina resulting in a unique material anisotropy that can be mathematically shown by its ABD-matrix. The Multistep method, similar to the one used by Eralp Demir et al. [8], is a stepwise method that uses a 3-step system to optimize the fiber paths of the object.

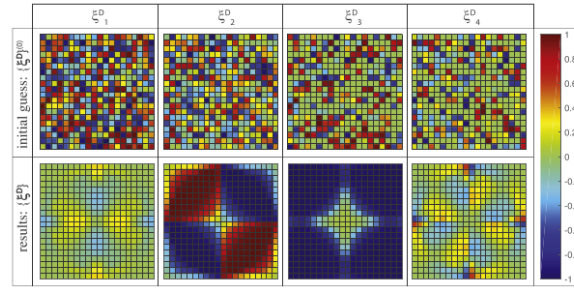


Figure 2.6: The Lamination Parameter method used by Eralp Demir et al. [8]

In the First step of the process, an optimizer is used on the 12 lamination parameters of the elements of the mesh which is a representation of the ABD Matrix as follows:

$$\begin{aligned}
 (V_{1A}, V_{2A}, V_{3A}, V_{4A}) &= \int_{-\frac{1}{2}}^{\frac{1}{2}} (\cos 2\theta, \sin 2\theta, \cos 4\theta, \sin 4\theta) dz \\
 (V_{1B}, V_{2B}, V_{3B}, V_{4B}) &= 4 \int_{-\frac{1}{2}}^{\frac{1}{2}} z (\cos 2\theta, \sin 2\theta, \cos 4\theta, \sin 4\theta) dz \\
 (V_{1D}, V_{2D}, V_{3D}, V_{4D}) &= 12 \int_{-\frac{1}{2}}^{\frac{1}{2}} z^2 (\cos 2\theta, \sin 2\theta, \cos 4\theta, \sin 4\theta) dz
 \end{aligned} \tag{2.17}$$

In these equations V_{iA} , V_{iB} , and V_{iD} represent the in-plane, coupling, and bending parameters of the matrix and z is the thickness axis of the laminate. These parameters can be further simplified in a few parameters using known lamination design rules, which can help designs against bending and coupling handling [28].

In the second step of the process, the fiber orientations are retrieved, from the lamination parameters, using different types of methods. There is not 1 correct solution, out of a fixed set of lamination parameters, for fiber orientation and stacking. Many approaches were designed to approach the lamination parameter. A closed-form inverse laminate design approach, used by Fukunaga et al. [11], which focuses on a direct analytic solution, or an evolutionary algorithmic method used by Autio M. et al. [4],

which uses an iterative algorithm to retrieve the optimal fiber orientations belonging to those lamination parameters.

The final step in the multistep method assimilates the fiber orientation mesh, created in the second step, into a full pathway streamlines design that can be implemented. This step has many articles referencing the streamline-based method, e.g. an article by Daniel P. [22], which uses the fiber angles as a velocity field where other manufacturing constraints are enforced. Other methods also include the B-Spline method [31], which uses control points to bend a flow field around them, and is then optimized for the fiber orientations, ensuring more smooth curvatures, which is good for manufacturing.

The multistep method has a better defined optimization problem that leads to a convex solution in the initial step, it removes the manufacturability from the initial step and focuses on structural performance first and then take care of manufacturability later on [1]. However, this can cause the manufacturable solution at the end of the multistep to not be the optimal fiber laying solution. This is where, instead, the DMO (Discrete Material Optimization) methods add manufacturability limits from the initial optimizer step.

2.3.2. DMO

The anisotropic properties of a laminate can also be directly retrieved from the stack up of fiber angles using e.g. the elastic constitutive matrix of each element's layer. The DMO methods use this to optimize directly from a possible discrete set of fiber orientations in each layer.

Some methods like a genetic algorithm method, used by An Haichao et al.[3], optimize the stacking sequence and implement strength constraints using the stresses and failures of the object. Others have also used anisotropic penalization, which penalizes the fiber angles that are causing unwanted behavior often e.g. fibers that are not along the stress directions [9]. Or even the EAOT, equivalent anisotropy orientation tensor method, which makes a directional tensor from the ply angle stacking sequence, and uses it like a compass to optimize the direction of the tensor, as shown by Zebei Mao [18].

Because the design variables highly increase in numbers, when additional layers or finer meshes are desired, these methods use either simplifications or penalization methods on discrete values, but resulting in a non-convex local-optima solution due to the nature of the problem. Newer methods, like NDFO (Normal Distribution Fiber Optimization) interpolations use a normal distributed weight system over the many candidate angles to retrieve 1 design variable per layer instead of many over different discrete angles [14].

$$D_{ij}^{mic}(\theta) = \sum_{P^c=1}^{N^{can}} \omega_{ij,P^c}(\theta) D_{ij,P^c} \quad (2.18)$$

$$\omega_{ij,P^c}(\theta) = e^{-\frac{(\theta - P^c)^2}{2(P^\sigma)^2}}, P^c \in 1, \dots, N^{can}, \theta \in [1 \ N^{can}]$$

The NDFO interpolation in the equation calculates the elastic constitutive matrix D_{ij}^{mic} of the element i and layer j using a weighted sum ω_{ij,P^c} of all the elastic constitutive matrices of the candidate angle D_{ij,P^c} with a weight in a normal distribution where the means are the different candidate angles P^c , the deviation is P^σ , and the weight is calculated using the location of the actual design variable's θ within the normal distribution for all candidates.

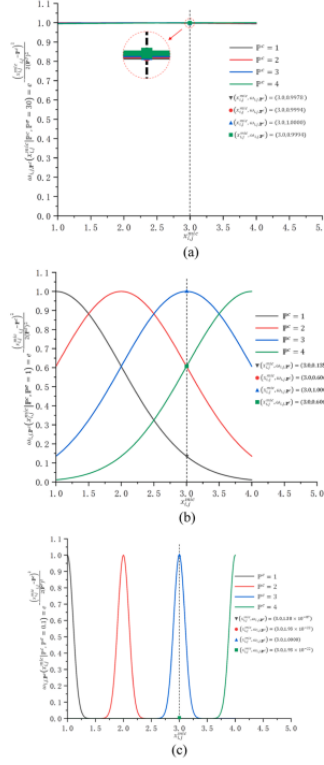


Figure 2.7: Graph of the weighing coefficients ω_{ij, P^σ} corresponding to different P^σ

This method uses a decreasing penalty parameter on the deviation of the distribution to decrease the early convergence of the system and thus decreasing chance of initially falling into local optima. However because of its general penalization scheme on all elements, some elements get blocked to early in the iterations or others are not settling in their values. That's where a proposed ANDFO (Adaptive Normal Distribution Fiber Optimization) scheme has been adapted for, by Zunyi Duan et al.[10].

The ANDFO scheme uses a similar initial setup as the NDFO scheme, where it implements an initial constant value for the penalty variable for k iterations, after which a linear decent is applied to converge the overall result into a more general optima. The ANDFO addition comes with the last adaptive stage where from a certain level of convergence the penalty parameter decreases adaptively unique to each element and layer, if that element hasn't achieved the density threshold ρ_{thre} or the normalized fiber weight threshold $\tilde{\omega}_{thre}$ yet. This way it converges the more difficult to converge elements as well by having their penalization unique to them.

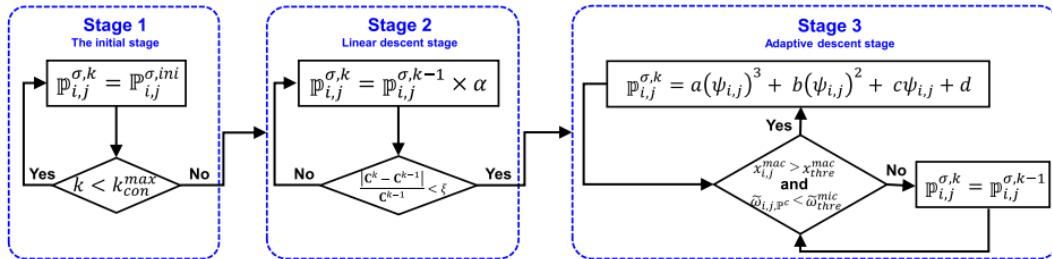


Figure 2.8: The 3 stages for the deviation penalty parameter P^σ during optimization in ANDFO

The Variable stiffness optimizer and topology optimizers both have a high quantity of research in their approaches to their respective problems and strategies in how to best handle certain complexities. The

next section will show some literature results of the combination methods of both optimizers to see how the system can result in 1 optimizer for composite laminate objects.

2.4. Combination Approaches

Multi-Step

The easiest approach to implement is the multistep method, it uses the decomposition strategy talked about in the strategy section of the structural optimization 2.1.2.

The method solves the optimizers sequentially by either first solving the topology optimizer and then the VSO or vice versa, e.g. Lee etc.[16] used a three-step optimizer where they initially retrieve the optimal topology after which in the second step the fiber orientation is optimized. Lastly, they refine the orientation for a more manufacturable result. The method is more computationally efficient and easier to implement than other approaches. However there are some missed optimization opportunities that are not considered within this method.

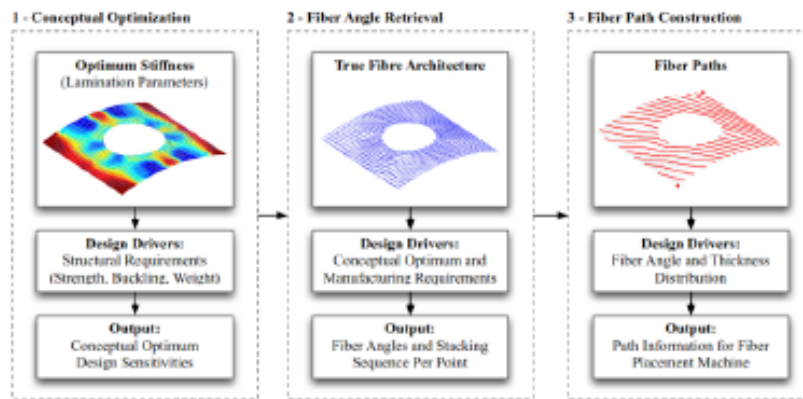


Figure 2.9: Multi-Step Methods using Lamination Parameters

Hierarchical

While the multistep goes about the optimization sequentially and thus separately, the hierarchical method uses a multiscale method inside an inner loop, it iterates over both optimizers with feedback responses. While it is not concurrent, as first an iteration of the macroscale is done after which an iteration of the microscale is done.

J.H.S. Almeida et al. [2], implemented a hierarchical method for optimizing topology and fiber orientation. They used a framework in which you do each optimization separately after the other. so in their framework specifically they initially do one increment with a topology optimizer after which they use the updated topology to do a step with the orientation optimizer. If after the second optimizer the system has converged it will take that current system as its optimal, if not it will redo the loop and start again with a next iteration of topology optimization.

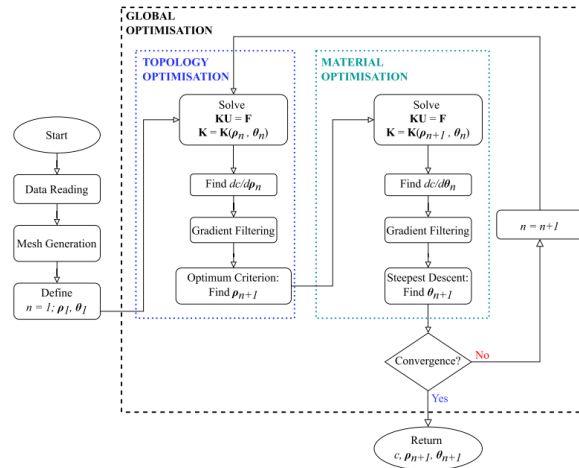


Figure 2.10: Example of a Hierarchical approach to the optimization used by J.H.S. Almeida et al. [2]

Concurrent optimization

While the hierarchical goes about the optimization sequentially in an outer loop, the concurrent optimization method runs both optimizers simultaneously. In this way it can truly be called a single optimizer instead of a smart framework with multiple optimizers.

In an article by Zunyi Duan et al. [10] using an ANDFO approach, a concurrent framework was used to simultaneously optimize the fiber orientations of the structure as well as optimize its topology.

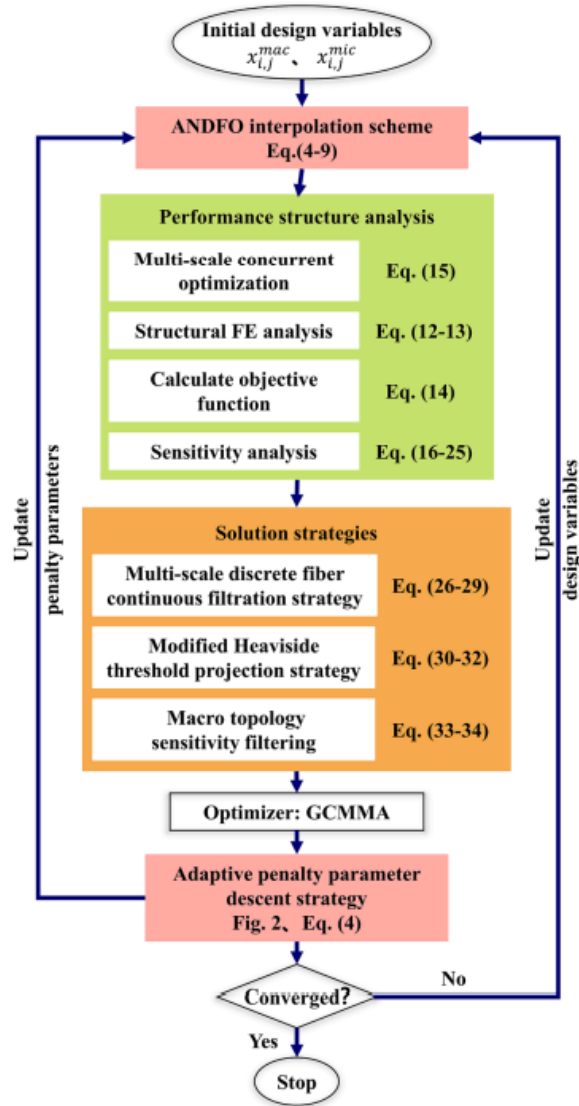


Figure 2.11: Example of a concurrent approach used by Zuryi Duan et al. [10]

Level-set with path optimization

The previously mentioned level-set method, in the topology optimization section 2.2.3, uses a evolving structural boundary to create its topology. This has been combine with fiber path methods like stream functions or velocity fields to obtain good manufacturability and smooth results. This approach comes with a high computational cost to use and a very complex implementation.

T. Xu et al. [33] used a combination of the level-set method where a level-set function for topology is used and a fiber path optimizer for VSO. They used them in a hierarchical framework from each other where an iteration was done on the level-set topology optimizer followed by an iteration on the fiber-path optimizer and those in an outer loop till convergence.

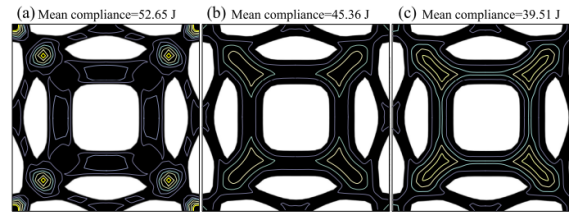


Figure 2.12: Example of the level-set approach with path optimization used by T.Xu et al. [33]

Conclusion

The different methods used in the literature study show that the variety of optimizers, optimization techniques and strategies are broad and open for bigger discussions. When focusing on the manufacturability aspect of the thesis, there are a few options that stand out as more suited for the constraints. In the case of the optimizer itself a GCMMA solver provides the best gradient based solution for more convex problems similar to the MMA solver its computational time is quite low as it brings the problem into a convex approximation after which the optimal result is sought. But the global convergence of the GCMMA above the MMA also provides an improvement on the chances of falling into a local optima as a more global solution is reached from it. The strategies section also showed that many strategies can aid in increasing the manufacturability, they can help guide the solution towards a more manufacturable solution as well as set foundational rules to solve micro structural problem that can show improved compliance without manufacturability. In topology optimization the complexity and CPU time of the system can drastically increase if more smooth topologies are desired for example in the level-set method. While this method, does provide a smoother result for manufacturability the increase in complexity for that slight smoothing improvement weighs against the improvement of its manufacturability in this case. The case of the density based model offers the continuity in parameter variables with a decreased CPU time, and with some combined strategies can show great manufacturable results. For VSO the manufacturability really shines with the DMO methods as it can be related directly on the fiber angle instead of having an optimized parameter result which then needs to be optimized separately for fibers. In particular, the new studies on ANDFO interpolation of the fiber angles have shown great results for manufacturability. For the combination approaches, a mix of the different approaches offers the best results as each approach has its advantages and disadvantages and can be used in a multi step breakdown each offering some level of hierarchical and some level of concurrent optimization with post processing.

3

Methodology

3.1. Finite Element Analysis

3.1.1. Classical Laminated Plate Theory

In this thesis CLPT (Classical Laminated Plate Theory) is implemented to model the behavior of the composite. This theory simplifies the physics of laminates into its in-plane and bending behavior, while neglecting transverse shear properties. This is computationally more favorable for assumptions of thin plates as used in this thesis, on top of having a pretty for thin laminates used in this thesis as shown by Odeh, Ali et al.[19]. With this simplified theory, the ABD matrix method can be used to define the behavior of the laminate. In CLPT the fiber orientation influences in-plane anisotropy through the ABD matrix.

ABD matrix

The ABD matrix represents the constitutive matrix of the whole laminate, in a relation form it gives the equation below 3.1.

$$\begin{bmatrix} \mathbf{N} \\ \mathbf{M} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varepsilon}^0 \\ \boldsymbol{\kappa} \end{bmatrix} \quad (3.1)$$

Here $\mathbf{N}$ and $\mathbf{M}$ represent the in-plane forces and bending moments on the laminate. $\boldsymbol{\varepsilon}^0$ represents the strain of the mid-plane of the laminate, and $\boldsymbol{\kappa}$ represents the curvature of the mid-plane.

Initially to retrieve the ABD-Matrix of a laminate in CLPT the constitutive matrix at each ply is required, this matrix is defined by the constitutive relation as such:

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_{12} \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{12} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_{12} \end{bmatrix} \quad (3.2)$$

In this formula the left side, σ_1, σ_2 respectively represent the stress in the principles directions 1 and 2 and τ_{12} represent the in-plane shear stress. On the right you have the constitutive matrix for the principle direction and the strain matrix for that principle direction, $\varepsilon_1, \varepsilon_2$ being the strains in the principle direction and γ_{12} is the shear strain. The matrix is derived from the material properties for the principle directions using the following equations:

$$Q_{11} = \frac{E_1}{1 - \nu_{12}\nu_{21}}, \quad Q_{22} = \frac{E_2}{1 - \nu_{12}\nu_{21}} \quad (3.3)$$

$$Q_{12} = \frac{\nu_{12}E_2}{1 - \nu_{12}\nu_{21}}, \quad Q_{66} = G_{12} \quad (3.4)$$

$$\nu_{21} = \nu_{12} \frac{E_2}{E_1} \quad (3.5)$$

When building the ABD matrix from the ply constitutive matrices, transformation need to be applied for their respective ply orientations. The orientated ply constitutive matrix can be obtained using the following equations.

$$m = \cos \theta, \quad n = \sin \theta \quad (3.6)$$

$$\begin{aligned} \bar{Q}_{xx} &= Q_{11}m^4 + 2(Q_{12} + 2Q_{66})m^2n^2 + Q_{22}n^4 \\ \bar{Q}_{yy} &= Q_{11}n^4 + 2(Q_{12} + 2Q_{66})m^2n^2 + Q_{22}m^4 \\ \bar{Q}_{xy} &= (Q_{11} + Q_{22} - 4Q_{66})m^2n^2 + Q_{12}(m^4 + n^4) \\ \bar{Q}_{xs} &= (Q_{11} - Q_{12} - 2Q_{66})m^3n + (Q_{12} - Q_{22} + 2Q_{66})mn^3 \\ \bar{Q}_{ys} &= (Q_{11} - Q_{12} - 2Q_{66})mn^3 + (Q_{12} - Q_{22} + 2Q_{66})m^3n \\ \bar{Q}_{ss} &= (Q_{11} + Q_{22} - 2Q_{12} - 2Q_{66})m^2n^2 + Q_{66}(m^4 + n^4) \end{aligned} \quad (3.7)$$

$$\bar{Q} = \begin{bmatrix} \bar{Q}_{xx} & \bar{Q}_{xy} & \bar{Q}_{xs} \\ \bar{Q}_{xy} & \bar{Q}_{yy} & \bar{Q}_{ys} \\ \bar{Q}_{xs} & \bar{Q}_{ys} & \bar{Q}_{ss} \end{bmatrix} \quad (3.8)$$

The ABD matrix follows then from the stacking sequence of the different ply constitutive matrices for each ply, and their respective location on the laminate thickness plane defined by z with the $z = 0$ plane being the mid-plane of the laminate.

$$\mathbf{A} = \sum_{k=1}^n \bar{Q}^{(k)} (z_k - z_{k-1}) \quad (3.9)$$

$$\mathbf{B} = \frac{1}{2} \sum_{k=1}^n \bar{Q}^{(k)} (z_k^2 - z_{k-1}^2) \quad (3.10)$$

$$\mathbf{D} = \frac{1}{3} \sum_{k=1}^n \bar{Q}^{(k)} (z_k^3 - z_{k-1}^3) \quad (3.11)$$

To retrieve the stiffness matrix using the ABD matrix in finite element analysis one must first define the elements and shapes used inside the FEA (Finite Element Analysis) model.

B matrix

The B-matrix is a representation of how the displacement inside the nodes of an element result in the strain values of that element by using the shape functions of the respective geometry of that element.

$$\epsilon = \mathbf{B}\mathbf{u} \quad (3.12)$$

In the thesis, quad elements are used in the mesh to solve the system, as they provide a better computational efficiency than higher order elements and is highly used in laminate composite modeling as it can be used to accurately represent the behavior of the element.

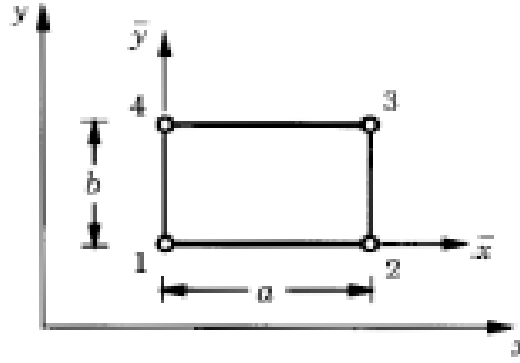


Figure 3.1: quad element representation in FEA

To retrieve the B matrix of this element, the shape functions are first required which represent the interpolation method to apply to any field on the element's node to retrieve its continuous element representation.

$$\mathbf{u} = \sum_{i=1}^n N_i u_i \quad (3.13)$$

In this equation the N_i is defined as the shape function of node i that is applied on the node's displacement u_i to retrieve the full displacement matrix of an element. The shape function in a quad element are defined by:

$$\begin{aligned} N_1(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 - \eta) \\ N_2(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 - \eta) \\ N_3(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 + \eta) \\ N_4(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 + \eta) \end{aligned} \quad (3.14)$$

In these equations for each node of the element the ξ and η represent the local coordinates in the range of $[-1, 1]$. Following this, the derivative of the shape functions in respect to the local coordinates are retrieved.

$$\begin{aligned} \frac{dN_1}{d\xi} &= -\frac{1}{4}(1 - \eta), & \frac{dN_2}{d\xi} &= \frac{1}{4}(1 - \eta), & \frac{dN_3}{d\xi} &= \frac{1}{4}(1 + \eta), & \frac{dN_4}{d\xi} &= -\frac{1}{4}(1 + \eta) \\ \frac{dN_1}{d\eta} &= -\frac{1}{4}(1 - \xi), & \frac{dN_2}{d\eta} &= -\frac{1}{4}(1 + \xi), & \frac{dN_3}{d\eta} &= \frac{1}{4}(1 + \xi), & \frac{dN_4}{d\eta} &= \frac{1}{4}(1 - \xi) \end{aligned} \quad (3.15)$$

The B matrix is made from the shape function derivative with respect to the global coordinate system x and y , for which the Jacobian is used:

$$\begin{bmatrix} \frac{dN_i}{dx} \\ \frac{dN_i}{dy} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{dN_i}{d\xi} \\ \frac{dN_i}{d\eta} \end{bmatrix} \quad (3.16)$$

The Jacobian is calculated from the derivatives of the shape functions in local coordinates and the coordinates of the nodes in global coordinates.

$$\mathbf{J} = \begin{bmatrix} \frac{dx}{d\xi} & \frac{dy}{d\xi} \\ \frac{dx}{d\eta} & \frac{dy}{d\eta} \end{bmatrix} = \sum_{i=1}^4 \begin{bmatrix} x_i \frac{dN_i}{d\xi} & y_i \frac{dN_i}{d\xi} \\ x_i \frac{dN_i}{d\eta} & y_i \frac{dN_i}{d\eta} \end{bmatrix} \quad (3.17)$$

After obtaining the shape functions derivative with respect to the global coordinate system, this can now be used to assemble the B matrix as follows:

$$\mathbf{B} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \dots & \frac{\partial N_n}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & \dots & 0 & \frac{\partial N_n}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_n}{\partial y} & \frac{\partial N_n}{\partial x} \end{bmatrix} \quad (3.18)$$

With these equations the element's stiffness matrix can be calculated next.

Element Stiffness Matrix

The element's stiffness matrix is mathematically defined, in CLPT, as the integral off the nodal forces out off the displacement of the nodes $\mathbf{B}^T \mathbf{Q} \mathbf{B}$, integrated over the element's volume dV .

$$\mathbf{K}_e = \int_{V_e} \mathbf{B}^T \mathbf{Q} \mathbf{B} dV \quad (3.19)$$

Because the laminate is made up of layers with constant thicknesses and constant constitutive matrices over each of those thicknesses as presented in equation 3.7, the out of plane integral can be set as a summation of the different layers as such:

$$\mathbf{K}_e = \sum_{l=1}^{n_{layers}} \int_{A_e} \mathbf{B}_e^T \mathbf{D}_{el} \mathbf{B}_e t_l dA \quad (3.20)$$

To simplify the analysis even more an approximation of the element's stiffness matrix can be done using Gauss points where the Gauss points for a quad element are

$$(\xi_I, \eta_J) = \left(\pm \sqrt{\frac{1}{3}}, \pm \sqrt{\frac{1}{3}} \right) \quad (3.21)$$

with Gauss weight of $W_I = W_J = 1$.

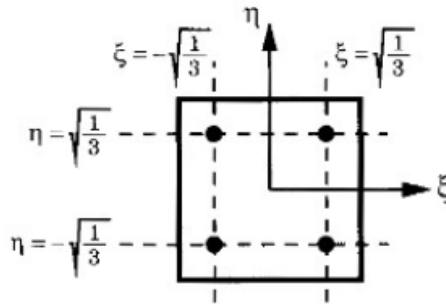


Figure 3.2: Gauss Integration Points in a quad element for FEA

When using the Gauss integration points on the stiffness matrix integral to approximate, it becomes

$$\mathbf{K}_e = \sum_{l=1}^{n_{layers}} \sum_{I=1}^2 \sum_{J=1}^2 \mathbf{B}_e^T(\xi_I, \eta_J) \mathbf{D}_{el}(\xi_I, \eta_J) \mathbf{B}_e(\xi_I, \eta_J) t_k \det(J(\xi_I, \eta_J)) W_I W_J \quad (3.22)$$

Boundary Conditions

The first part of the BCs are the known displacement defined as u_k . They represent the clamping and other fixed motions of the problem. Each of these displacements are saved in the known displacement vector u_k and their respective DOFs are saved.

For fixed node DOF this is just adding the displacement to the corresponding node DOF, but for the edge displacements the corresponding nodes for that edge is first retrieved and then these nodes DOF displacements are set to the edge displacement that was set.

Next to displacement boundary conditions you also have force boundary conditions which is implemented in the next section.

force vector

In this thesis for optimization only point loads and distributed surface loads are considered, body force like gravity are neglected as they are applied on the volume which for a CLPT is greatly diminished by its thin thickness.

$$f_e = \int_{V_e} \mathbf{N}^T \mathbf{b} dV + \int_{S_e} \mathbf{N}^T \mathbf{t} dS + f_e^{point} \quad (3.23)$$

Point loads are implemented by simply setting their value in the correct location in the force matrix, which in our CLPT case is often registered to the corresponding node and it's corresponding dof.

$$f_e = f_e^{point} \quad (3.24)$$

For distributed forces or tractions, 2 cases are considered. First a distributed force on an edge or line of an element, line traction, often integrated over the line. Secondly, a surface traction/pressure over a surface which is integrated in response over the whole applicable surface. However, as the surface traction and or pressure influence the transverse displacement it is not considered in this case as we use a simplified in-plane model, which do not include the transverse DOFs.

The force resultant from a line traction p_{xy} , can be calculated using the integral as follows

$$f_e = \int_{\Gamma_e} \mathbf{N}^T \mathbf{p}_{xy} d\Gamma \quad (3.25)$$

Here the $\mathbf{N}$ represents the shape function, p the pressure or edge traction in N/mm and $d\Gamma$ is the loaded line. After Gauss interpolation this becomes:

$$\begin{aligned} f_e &= \sum_{J=1}^2 \mathbf{N}^T(\xi_e, \eta_J) \mathbf{p}_{xy}(\eta_J) \det(J_\Gamma(\eta_J)) W_J, & \text{with } \det(J_\Gamma) &= \frac{l}{2}, \xi_e = [-1/1] \\ f_e &= \sum_{I=1}^2 \mathbf{N}^T(\xi_I, \eta_e) \mathbf{p}_{xy}(\eta_e) \det(J_\Gamma(\xi_I)) W_I, & \text{with } \det(J_\Gamma) &= \frac{l}{2}, \eta_e = [-1/1] \end{aligned} \quad (3.26)$$

Here the initial function represents the edge load cases where the ξ edge is selected with either the left or right edge respectively $[-1/1]$. The second function follows the same case but for the top and bottom edges representing η .

Now that the force vector and the stiffness matrices have been formed the problem properties and compliance can be calculated.

strains and stresses

Initially, the global stiffness matrix is assembled by adding all the element stiffness matrices obtained from eq 3.22. Secondly, the known displacement DOFs, referenced by subscript k, and the unknown DOFs, referenced by subscript u, are separated from all the matrices in the global equilibrium equation:

$$\mathbf{K}\mathbf{u} = \mathbf{F}$$

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{uk} \\ \mathbf{K}_{ku} & \mathbf{K}_{kk} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_k \end{bmatrix} = \begin{bmatrix} \mathbf{F}_u \\ \mathbf{F}_k \end{bmatrix} \quad (3.27)$$

To Solve : $\mathbf{K}_{uu}\mathbf{u}_u = \mathbf{F}_u - \mathbf{K}_{uk}\mathbf{u}_k$

Solving the system above for the unknown displacements and assembling them together with the known displacement returns then the displacement matrix $\mathbf{u}$. The strains and stresses can then be calculated in order from the displacement matrix using the equations 3.12 and 3.2, with Gauss point the central point of the element if you wish for a general representation of the strain and stresses in that element.

Compliance

The compliance C is a value representing the deformation of a structure under a given load. A high compliance thus means that the structure deforms a lot, while a low compliance means that the structure is stiff. It is mathematically calculated by the force and displacement vector as follows:

$$C = \mathbf{F}^T \mathbf{u} \quad (3.28)$$

It can also be represented by the relation with strain energy U and the stiffness matrix $\mathbf{K}$ using eq 3.27 as such:

$$U = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u}$$

$$C = \mathbf{u}^T \mathbf{K} \mathbf{u} = 2U \quad (3.29)$$

So to conclude the FEA model uses a CLPT method with an ABD matrix on an element basis and focus on in plane loads. It also implements a Gauss integration over the quad elements for a more integrated element stiffness matrix and to see the change in displacement and stress and strain over the element. Also the boundary conditions and compliance are fitted into the methodology. With the setup of the FEA ready the topology optimization is constructed next with the rules of the model.

3.2. Topology Optimization

Topology optimization, as mentioned before in 2.2, is a structural optimization method that retrieves where a structure should have material for an optimal objective function results, in our case that refers to compliance. The three methods mentioned in that section, density-based, BESO, or level-set, are all good approaches for topology optimization. However, while the level-set method shines in providing a more manufacturable component, it comes with high complexity into coupling it with fiber and high computational cost, which is not desired for this thesis. The BESO method is easy to understand and implement but doesn't allow for continuous gradient-based analysis and is more complex for VSO coupling. The density based gradient, SIMP in specific, has a simple setup that used a continuous design variable that can be implemented with our FEA analysis.

3.2.1. SIMP

There are two use cases that can be implemented with the SIMP method, the element-based and layer-based method. While the element-based method, where the density is applied on the whole element, drastically decreases the number of design variables to 1 per element. However, this method does not allow interchangeable densities throughout the layers which leaves out interesting cases where there might be more layers in certain locations than others. That is where the layer based method comes in,

where the density is applied on each layer separately. It provides much more control over the layers that are necessary for the structure in each element but comes with higher computational costs.

On top of the layer based method a penalty power system p is used with the SIMP density to push the values of the density ρ toward 0 or 1.

In CLPT it is defined as:

$$\mathbf{D}_{el}^{SIMP}(\rho) = \rho_{el}^p \mathbf{D}_{el} \quad (3.30)$$

Where $\mathbf{D}_{el}^{SIMP}$ represents the layer's constitutive matrix after application of the density SIMP model for element e and layer l .

3.2.2. Problem

The problem of the topology is defined as minimizing the objective function F , represented as the compliance of the structure C , with reference to the densities ρ of the elements in the structure, with a constraint on the total density used inside the structure limited by a resource constraint R .

$$\begin{aligned} \min F(\rho) &= \min C(\rho), \\ \text{with } \int_{\Omega} \rho dV &\leq R, \rho \in [0, 1] \end{aligned} \quad (3.31)$$

As stated before in 3.1.1, the compliance equilibrium is defined as twice the strain energy or as equation 3.28. The minimum of the compliance function with respect to ρ , can then also be written as:

$$\begin{aligned} \frac{dC(\rho)}{d\rho} &= \frac{d(\mathbf{F}^T \mathbf{u}(\rho))}{d\rho} \\ \frac{dC(\rho)}{d\rho} &= \mathbf{F}^T \frac{d\mathbf{u}(\rho)}{d\rho} \end{aligned} \quad (3.32)$$

From the equilibrium equation 3.27, we can take the derivative as such:

$$\frac{d(\mathbf{K}\mathbf{u})}{d\rho} = \frac{d\mathbf{F}^T}{d\rho} \quad (3.33)$$

The force vector does not change with density, so $\frac{d\mathbf{F}^T}{d\rho} = 0$.
Then after applying the product rule:

$$\begin{aligned} \frac{d(\mathbf{K}\mathbf{u})}{d\rho} &= 0 \\ \frac{d\mathbf{K}}{d\rho} \mathbf{u} + \mathbf{K} \frac{d\mathbf{u}}{d\rho} &= 0 \\ \frac{d\mathbf{K}}{d\rho} \mathbf{u} &= -\mathbf{K} \frac{d\mathbf{u}}{d\rho} \end{aligned} \quad (3.34)$$

Adding this equation and the equilibrium equation 3.27 to the sensitivity equation 3.32, we get

$$\begin{aligned} \frac{dC(\rho)}{d\rho} &= \mathbf{u}^T \mathbf{K} \frac{d\mathbf{u}(\rho)}{d\rho} \\ \frac{dC(\rho)}{d\rho} &= -\mathbf{u}^T \frac{d\mathbf{K}(\rho)}{d\rho} \mathbf{u} \end{aligned} \quad (3.35)$$

Next, is the derivation of the derivative of the stiffness matrix with respect to ρ . The global stiffness matrix is a summation of all the different element stiffness matrices, where each stiffness matrix only changes due to its own design variable ρ , the others are not affected leading to:

$$\begin{aligned}\frac{d\mathbf{K}(\rho_e)}{d\rho_e} &= \sum_{i=0}^{N_{elem}} \frac{d\mathbf{K}_i(\rho_e)}{d\rho_e} \\ \frac{d\mathbf{K}_i(\rho_e)}{d\rho_e} &= 0 \quad \text{for } i \neq e \\ \frac{d\mathbf{K}(\rho_e)}{d\rho_e} &= \frac{d\mathbf{K}_e(\rho_e)}{d\rho_e}\end{aligned}\tag{3.36}$$

Using equation 3.20 to fill the element stiffness matrix and substituting the SIMP constitutive matrix gives

$$\begin{aligned}\frac{d\mathbf{K}(\rho_e)}{d\rho_e} &= \frac{d\mathbf{K}_e(\rho_e)}{d\rho_e} \\ \frac{d\mathbf{K}(\rho_e)}{d\rho_e} &= \frac{d}{d\rho_e} \left(\sum_{l=1}^{n_{layers}} \int_{A_e} \mathbf{B}_e^T \mathbf{D}_{el}^{SIMP}(\rho_e) \mathbf{B}_e t_l dA \right)\end{aligned}\tag{3.37}$$

Because the B matrix $\mathbf{B}_e$ and the thickness t_l have no dependencies on the density, and the density variable for a layer does not have an effect on the constitutive matrix of another layer, it can be deduced to the following equation:

$$\frac{d\mathbf{K}(\rho_{el})}{d\rho_{el}} = \int_{A_e} \mathbf{B}_e^T \frac{d\mathbf{D}_{el}^{SIMP}(\rho_{el})}{d\rho_{el}} \mathbf{B}_e t_l dA\tag{3.38}$$

Now from the previous definition of the SIMP constitutive matrix of a layer 3.30, we can retrieve the gradient as such:

$$\begin{aligned}\frac{d\mathbf{D}_{el}^{SIMP}(\rho_{el})}{d\rho_{el}} &= \frac{d}{d\rho_{el}} (\rho_{el}^p \mathbf{D}_{el}) \\ \frac{d\mathbf{D}_{el}^{SIMP}(\rho_{el})}{d\rho_{el}} &= p\rho_{el}^{p-1} \mathbf{D}_{el}\end{aligned}\tag{3.39}$$

Now implementing this equation back into eq 3.38 and 3.35 gives:

$$\frac{dC(\rho_{el})}{d\rho_{el}} = -\mathbf{u}^T \int_{A_e} \mathbf{B}_e^T p\rho_{el}^{p-1} \mathbf{D}_{el} \mathbf{B}_e t_l dA \mathbf{u}\tag{3.40}$$

After applying gauss we get:

$$\frac{dC(\rho_{el})}{d\rho_{el}} = -\mathbf{u}^T \sum_{I=1}^2 \sum_{J=1}^2 \mathbf{B}_e^T(\xi_I, \eta_J) p\rho_{el}^{p-1} \mathbf{D}_{el} \mathbf{B}_e(\xi_I, \eta_J) t_l \det(J(\xi_I, \eta_J)) W_I W_J \mathbf{u}\tag{3.41}$$

This can also be rewritten to be easier made as:

$$\begin{aligned}
\frac{dC(\rho_{el})}{d\rho_{el}} &= -\mathbf{u}^T \frac{p}{\rho_{el}} \sum_{I=1}^2 \sum_{J=1}^2 \mathbf{B}_e^T(\xi_I, \eta_J) \rho_{el}^p \mathbf{D}_{el} \mathbf{B}_e(\xi_I, \eta_J) t_l \det(J(\xi_I, \eta_J)) W_I W_J \mathbf{u} \\
\frac{dC(\rho_{el})}{d\rho_{el}} &= -\mathbf{u}^T \frac{p}{\rho_{el}} \sum_{I=1}^2 \sum_{J=1}^2 \mathbf{B}_e^T(\xi_I, \eta_J) \mathbf{D}_{el}^{SIMP} \mathbf{B}_e(\xi_I, \eta_J) t_l \det(J(\xi_I, \eta_J)) W_I W_J \mathbf{u} \\
\frac{dC(\rho_{el})}{d\rho_{el}} &= -\mathbf{u}^T \frac{p}{\rho_{el}} \mathbf{K}_{el}(\rho_{el}) \mathbf{u}
\end{aligned} \tag{3.42}$$

With the topology problem there is also a need for a constraint gradient, which comes in play with the volume gradient limiting the amount of material used by the problem.

3.2.3. Volume Gradient

For the GCMMA solver used in this thesis, the Volume gradient is also necessary to retrieve the constraint's approximation function. The constraint function can be seen as the application of the density of an element on that elements volume respecting a certain resource constraint or volume fraction, thus giving:

$$g(\rho) = \frac{1}{V_{tot}} \sum_{l=1}^{N_{layers}} \sum_{e=1}^{N_{elem}} \rho_{el} V_{el} - f_V \leq 0 \tag{3.43}$$

Where $g(\rho)$ represents the constraint function which should be lower than or equal to 0. The V_{tot} variable is the total volume of the design domain, V_{el} is each element layer's volume. The resource or volume fraction in this case is represented by a volume fraction f_V . For the GCMMA a gradient with reference to the design variable ρ_{el} is found by the derivative of the constraint function:

$$\frac{dg(\rho)}{d\rho_{el}} = \frac{V_{el}}{V_{tot}} = G_V \tag{3.44}$$

3.3. Fiber Optimization

The fiber optimization, or VSO, can also be used with different methods to retrieve a good optimized result. The multistep lamination parameter approach provides a good simple system to execute, however it does lack in manufacturable area as it is only adapted in the system in the post-processing part which eliminates the effect of the manufacturability within the optimization itself. So a DMO approach for this thesis is more useful as it gives the optimizer direct control of the orientation of the fiber. Within the DMO approaches, our thesis focuses on the ANDFO approach as it provides a more refined and adaptive system for the DMO method.

3.3.1. NDFO

The ANDFO follows the same system as NDFO when it comes to the design variable dependency inside the system. As previously shown in section 2.3.2, the fiber orientation is used in a normal distribution weighting system where the design variable is between $[1, N_{cand}]$, where N_{cand} is the amount of candidate fiber angles chosen for the optimization. The implementation of the fiber angle is done through the layer constitutive matrix where equation 2.18 is used.

$$\begin{aligned}
\mathbf{D}_{el}^{NDFO}(\theta) &= \sum_{P^c=1}^{N_{can}} \frac{\omega_{el, P^c}(\theta)}{\sum_{P^c=1}^{N_{cand}} \omega_{el, P^c}(\theta)} \mathbf{D}_{el, P^c}, \\
\omega_{el, P^c}(\theta) &= e^{-\frac{(\theta - P^c)^2}{2(P^c)^2}}, \quad P^c \in 1, \dots, N_{cand}, \quad \theta \in [1, N_{cand}]
\end{aligned} \tag{3.45}$$

Here $\mathbf{D}_{el}^{NDFO}$ represents the constitutive matrix of the layer where the NDFO interpolation has been applied to the actual constitutive matrix for each candidate angle $\mathbf{D}_{el, P^c}$, and which is later applied to

topology optimization for density applications. θ is a representation of the orientation of the fiber α , where the value corresponds to:

$$\theta = \frac{N_{\text{cand}}}{180}(\alpha + 90), \quad \theta = [1, N_{\text{cand}}] \quad \alpha = \left[\frac{180}{N_{\text{cand}}} - 90, +90\right] \quad (3.46)$$

The P^c refers to the candidate orientations which are the discrete values from $1, \dots, N^{\text{cand}}$. It refers to the mean value of each weight's normal distribution. The standard deviation of each distribution is represented by P^σ . So in the equation for the weight the value is retrieved by checking where the design variable θ currently lies in all the normal distribution of corresponding to each candidate angle.

3.3.2. Problem

The problem for the VSO follows a similar pattern as the one for the topology optimization, where the objective function is the compliance C for which the minimum is optimized with reference to the design variable θ of element e and layer l . The same initial conditions hold to result in the following equation

$$\frac{dC(\theta_{el})}{d\theta_{el}} = -\mathbf{u}^T \frac{d\mathbf{K}(\theta_{el})}{d\theta_{el}} \mathbf{u} \quad (3.47)$$

The same element and layer dependencies hold, as well as the B-matrix and thickness dependencies, resulting in the following:

$$\frac{dK(\theta_{el})}{d\theta_{el}} = \int_{A_e} \mathbf{B}_e^T \frac{d\mathbf{D}_{el}^{\text{SIMP}}(\theta_{el})}{d\theta_{el}} \mathbf{B}_e t_l dA \quad (3.48)$$

Here the SIMP constitutive matrix can be written down as:

$$\frac{d\mathbf{D}_{el}^{\text{SIMP}}(\theta_{el})}{d\theta_{el}} = \frac{d}{d\theta_{el}} (\rho_e^p \mathbf{D}_{el}^{\text{NDFO}}) \quad (3.49)$$

Because the density is not affected by the fiber orientation we can pass on the derivation to the NDFO constitutive matrix:

$$\frac{d\mathbf{D}_{el}^{\text{SIMP}}(\theta_{el})}{d\theta_{el}} = \rho_e^p \frac{d\mathbf{D}_{el}^{\text{NDFO}}}{d\theta_{el}} \quad (3.50)$$

Filling in the equation for the NDFO constitutive matrix and knowing that the constitutive matrix $\mathbf{D}_{el, P^c}$ is not dependent on θ gives:

$$\frac{d\mathbf{D}_{el}^{\text{NDFO}}}{d\theta_{el}} = \sum_{P^c=1}^{N_{\text{can}}} \frac{d}{d\theta_{el}} \left(\frac{\omega_{el, P^c}(\theta)}{\sum_{P^c=1}^{N^{\text{cand}}} \omega_{el, P^c}(\theta)} \right) \mathbf{D}_{el, P^c} \quad (3.51)$$

Applying the quotient rule:

$$\frac{d}{d\theta_{el}} \left(\frac{\omega_{el, P^c}(\theta)}{\sum_{P^c=1}^{N^{\text{cand}}} \omega_{el, P^c}(\theta)} \right) = \frac{\frac{d}{d\theta_{el}}(\omega_{el, P^c}(\theta)) \sum_{P^c=1}^{N^{\text{cand}}} \omega_{el, P^c}(\theta) - \omega_{el, P^c}(\theta) \left(\sum_{P^c=1}^{N^{\text{cand}}} \frac{d}{d\theta_{el}} \omega_{el, P^c}(\theta) \right)}{\left(\sum_{P^c=1}^{N^{\text{cand}}} \omega_{el, P^c}(\theta) \right)^2} \quad (3.52)$$

Separately:

$$\begin{aligned}
\frac{d}{d\theta_{el}}(\omega_{el, P^c}(\theta)) &= \frac{d}{d\theta_{el}} \left(e^{-\frac{(\theta - P^c)^2}{2(P^\sigma)^2}} \right) \\
\frac{d}{d\theta_{el}}(\omega_{el, P^c}(\theta)) &= -\frac{\theta - P^c}{(P^\sigma)^2} e^{-\frac{(\theta - P^c)^2}{2(P^\sigma)^2}} \\
\frac{d}{d\theta_{el}}(\omega_{el, P^c}(\theta)) &= -\frac{\theta - P^c}{(P^\sigma)^2} \omega_{el, P^c}(\theta)
\end{aligned} \tag{3.53}$$

3.3.3. Adaptive Descent Strategy

As previously explained and shown in equation 3.45, the ANDFO method uses a deviation penalty parameter P^σ on the weights to push the fiber orientation design variable towards the candidates angles set. However, to initially allow the design variable to move around and explore the full range of fiber orientations the penalty parameter is set to a large enough number to allow that initial movement. This is then followed by a linear decreasing strategy after certain amount of iterations to start locking in the fiber orientation into it's respective candidate angles. This linear decent strategy for the penalty parameter allows the design to settle down and converge to these discrete candidate angle options. This strategy however still allows for some locations in the object to have local non-convergences and oscillatory behavior of their local compliance levels. That's why an adaptive strategy is applied afterward, to push the local element behaviors also towards their own local convergences. To check the local convergence of each element the elements artificial density and artificial weighting coefficient is checked for convergence by means of a threshold value. The penalty parameter, if the element does not comply with these thresholds, is then adapted to by means of using the convergence gap between the candidate angles and the design variable orientation to make the design variable lock into their respective candidate angles more. To achieve this a polynomial is used that follows a pattern for the penalty parameter in respect to the convergence gap as shown by Zunyi Duan et al. [10]. The full penalization decent strategy is shown below in fig 3.3.

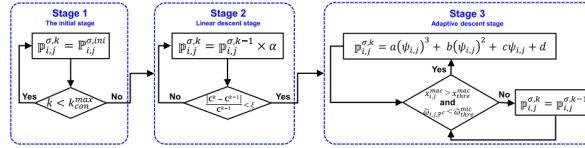


Figure 3.3: The 3 decent stages for the deviation penalty parameter

3.3.4. Curvature Constraint

To apply a well definable manufacturing limitation, on the optimization, a curvature constraint is added which pushes the optimization problem of fiber steering towards a solution where the curvature between elements is constraint by the minimum steering curvature.

The curvature in the model is mathematically defined as the gradient of the spatial field of the fiber angle orientations, $\kappa = ||\nabla\theta||$, as shown by Van Campen, J.M.J.F. et al. [27]. To fiber orientation is defined in our model per element which makes the evaluation of the curvature to be defined as the magnitude sum of the change in fiber orientation from on element to it's neighbor over their centroids distances.

$$\kappa = \frac{1}{2} \sqrt{\frac{1}{N_i} \sum_{j \in \mathcal{N}(i)} \left(\frac{\phi_i - \phi_j}{d_{ij}} \right)^2}, \quad \text{where } \phi = 2\left(\frac{\pi}{N_{cand}}\theta - \frac{\pi}{2}\right) \tag{3.54}$$

The fiber orientation is doubled to get ϕ because the fiber orientation is periodic in π and not 2π . The curvature is then compared against the maximum curvature, $\kappa^{max} = \frac{1}{r^{min}}$, which is the inverse of the minimum steering radius, $\frac{\kappa}{\kappa^{max}}$. The constraint function g is then defined as a sum of all curvature fractions of the elements to a certain penalty power p that can increase the sensitivity to outliers that neglect the curvature constraint. To make sure that the curvature of each element stays physically interesting the density of that element is also applied on the curvature, this allows the curvatures constraint

function to be mostly dependent off actual material presentable elements. Thus, the final constraint function becomes:

$$g = \left(\frac{1}{N_{elem}} \sum_{e \in N_{elem}} \left(\frac{\rho_e \kappa}{\kappa^{max}} \right)^p \right) - 1 \quad (3.55)$$

To find the gradient of the constraint function with respect to the fiber orientation 3 product rules are applied where:

$$\begin{aligned} \frac{dg}{d\theta} &= \frac{dg}{d\kappa} \frac{d\kappa}{dS} \frac{dS}{d\phi} \frac{d\phi}{d\theta} \\ S &= \sum_{j \in \mathcal{N}(i)} \left(\frac{\phi_i - \phi_j}{d_{ij}} \right)^2 \\ \frac{dg}{d\kappa_e} &= \frac{p}{N_{elem}} \frac{\rho_e}{\kappa^{max}} \left(\frac{\rho_e \kappa_e}{\kappa^{max}} \right)^{p-1} \\ \frac{d\kappa_e}{dS_e} &= \frac{1}{4N_i \kappa_e} \\ \frac{dS_e}{d\phi_i} &= 2 \sum_{j \in \mathcal{N}(e)} \frac{\phi_e - \phi_j}{d_{ej}^2} \cdot M_{ej} \\ \frac{d\phi_i}{d\theta_i} &= 2 \frac{\pi}{N_{cand}} \\ G_\kappa &= \frac{p}{N_{elem}} \frac{\rho_e}{\kappa^{max}} \left(\frac{\rho_e \kappa_e}{\kappa^{max}} \right)^{p-1} \frac{1}{N_i \kappa_e} \sum_{j \in \mathcal{N}(e)} \frac{\phi_e - \phi_j}{d_{ej}^2} \cdot M_{ej} \frac{\pi}{N_{cand}} \end{aligned} \quad (3.56)$$

The optimizers provide the model now with the routes to drive the problem towards an optimal solution. Next few strategies are applied that provide certain guidances in the route towards the optimal solution.

3.4. Solution Strategies

The optimizers implemented have an affinity for using checkerboard patterns or similar patterns in the fiber angles where two aligning elements can have a high difference in angle. To solve these occurrences, strategies and filters are implemented as mentioned in 2.1.3

3.4.1. Density Filter

For the topology, the density design variables can be filtered using the equation as shown in the strategy section giving:

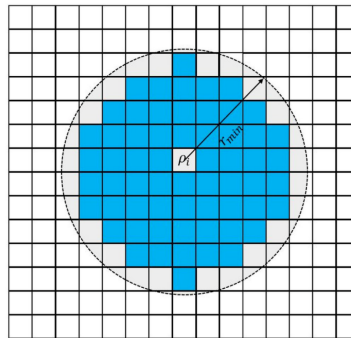


Figure 3.4: Density Filter on Topology Optimization

$$\tilde{\rho}_{el} = \frac{\sum_{k \in N(e)} \omega_k \rho_{kl}}{\sum_{k \in N(e)} \omega_k} \quad (3.57)$$

The $N(e)$ represents the set of elements within the parametrized boundary defined as the neighbors of element e . ω_{kl} refers then to the weight of neighboring element k , determined by its distance explained later. ρ_{kl} refers to the density of same element as the weight, k , at the same layer of the to be derived, density variable ρ_{el} .

The neighbors are determined by the elements whose central point lays within the circle created by a radius parameter R_ρ . The weight of that neighboring element is then calculated through the normal of the distance between the central point of the neighboring element and the boundary set by the circle of R_ρ , represented as follows:

$$\omega_k = \frac{R_\rho - R}{R_\rho}, \quad \text{where } R = \text{dist}(C_e, C_k) \quad \text{for } R \leq R_\rho \quad (3.58)$$

In this weight equation C_e and C_k represent the central points of the focused element e and it's neighbor k .

The function is also implemented into the sensitivity analysis as now the gradient is taken on the filtered design variable, and thus the product rule can be used as such:

$$\frac{dC}{d\rho_{el}} = \sum_{k \in N(e)} \frac{dC}{d\tilde{\rho}_{kl}} \frac{d\tilde{\rho}_{kl}}{d\rho_{el}} \quad (3.59)$$

Where $\frac{dC}{d\tilde{\rho}_{el}}$ is the gradient equation 3.42 previously defined in section 3.3.2 applied on the $\tilde{\rho}_{el}$ calculated from the filter equation 3.57. $\frac{d\tilde{\rho}_{el}}{d\rho_{el}}$ represents the change in the $\tilde{\rho}_{el}$ over the actual ρ_{el} . Which is calculated as such:

$$\frac{d\tilde{\rho}_{kl}}{d\rho_{el}} = \frac{d}{d\rho_{el}} \left(\frac{\sum_{k \in N(e)} \omega_k \rho_{kl}}{\sum_{k \in N(e)} \omega_k} \right) \quad (3.60)$$

This only holds a value when they lie in the same element and thus can be simplified to:

$$\frac{d\tilde{\rho}_{kl}}{d\rho_{el}} = \frac{\omega_e}{\sum_{k \in N(e)} \omega_k} \quad (3.61)$$

3.4.2. Fiber Filter

For the fiber orientation filtering both the design variable and sensitivity filtering are applied, as the design variable filtering will make sure that fiber orientations of neighboring elements have similar orientations. The sensitivity filtering will help more with making sure that these neighboring elements change together after every iteration, this can be understood by saying that if the optimizer want to change a certain element's fiber orientation he would have to adjust the whole area around that element at a certain level.

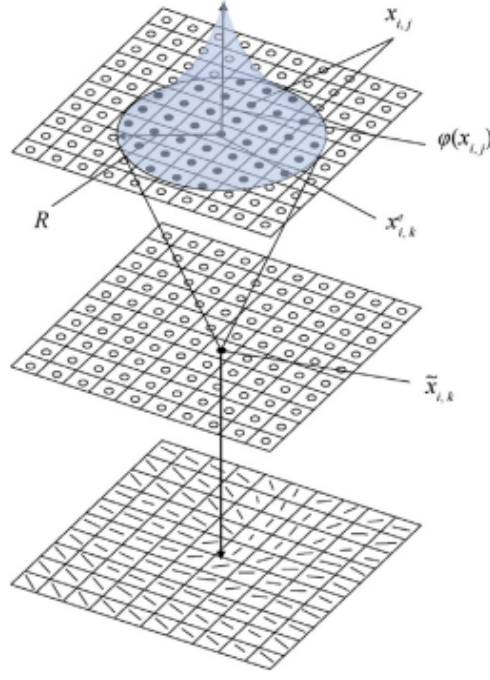


Figure 3.5: nonlinear Filter Strategy

The importance of the fiber orientation filtering is highest near the element to make sure that the next fiber in line follows the same direction but not get influenced too much by the fiber after that and thus limiting the movement of the overall fiber in general. That is why a higher order weight function is applied on the fiber orientation design variable. On top of this higher order an influence of the density of an element is also added to the filtering to make sure that the fiber orientation filtering only gets influenced by elements whom actually have a density prescribing material placement and not voids. Both these considerations result in the following equation:

$$\tilde{\theta}_{el}(\rho) = \frac{\sum_{k \in N(e)} \omega_k \rho_{kl} \theta_{kl}}{\sum_{k \in N(e)} \omega_k \rho_{kl}} \quad (3.62)$$

Where the weight is represented as:

$$\omega_k = \left(\frac{R_\theta - R}{R_\theta} \right)^p, \quad \text{where } R = \text{dist}(C_e, C_k) \quad \text{for } R \leq R_\theta \quad (3.63)$$

For the topology optimization the fiber filter is not assumed applicable because when the density is optimized the fiber orientation is assumed fixed at that iteration. So while the density is used as a variable in the filter, there is no adaptation on the density sensitivity for topology optimization.

However, for the sensitivity of the fiber orientation the filter is adapted as it helps with setting how the fiber orientation should change knowing the filter is used. This leads to the following equation derived similarly as before:

$$\frac{dC}{d\theta_{el}} = \sum_{k \in N(e)} \frac{dC}{d\tilde{\theta}_{kl}} \frac{d\tilde{\theta}_{kl}}{d\theta_{el}} \quad (3.64)$$

Where $\frac{dC}{d\theta_{el}}$ is the gradient equation previously defined in section 3.3 applied on $\tilde{\theta}_{el}$. $\frac{d\tilde{\theta}_{el}}{d\theta_{el}}$ represents the change in the filtered orientation over the actual orientation. Which is calculated as such:

$$\frac{d\tilde{\theta}_{kl}}{d\theta_{el}} = \frac{d}{d\theta_{el}} \left(\frac{\sum_{o \in N(k)} \omega_o \rho_{ol} \theta_{ol}}{\sum_{o \in N(k)} \omega_o \rho_{ol}} \right) \quad (3.65)$$

Following the same line of thought as the previous calculations in the density filter, the equation only holds a value for the same element.

$$\frac{d\tilde{\theta}_{kl}}{d\theta_{el}} = \frac{\omega_e \rho_{el}}{\sum_{o \in N(k)} \omega_o \rho_{ol}} \quad (3.66)$$

As for the sensitivity version of the filter that is used for smoothing the gradients of neighboring element uses the same weighting as the design variable filter and can be defined as follows:

$$\frac{d\tilde{C}}{d\theta_{el}}(\rho) = \frac{\sum_{k \in N(e)} \omega_k \rho_{kl} \frac{dC}{d\theta_{kl}}}{\sum_{k \in N(e)} \omega_k \rho_{kl}} \quad (3.67)$$

The density filter can also be used inside the fiber orientation filtering. It would mean that the more an element is in a dense region of multiple dense neighbors the more the fiber orientation filtering will take place as well. This way it can help reduce fiber filtering for less dense regions where the control should be smaller.

3.4.3. Substitution

Topology Optimization

For Topology optimization substituting the filters inside the gradient gives:

$$\frac{dC}{d\tilde{\rho}_{kl}} = -\mathbf{u}^T \sum_{I=1}^2 \sum_{J=1}^2 \mathbf{B}_k^T(\xi_I, \eta_J) p \tilde{\rho}_{kl}^{p-1} \mathbf{D}_{kl} \mathbf{B}_k(\xi_I, \eta_J) t_l \det(J(\xi_I, \eta_J)) W_I W_J \mathbf{u} \quad (3.68)$$

Then substituting eq 3.68 in eq 3.59 gives:

$$G_1 = \frac{dC}{d\rho_{el}} = \sum_{k \in N(e)} \left(-\mathbf{u}^T \sum_{I=1}^2 \sum_{J=1}^2 \mathbf{B}_k^T(\xi_I, \eta_J) p \tilde{\rho}_{kl}^{p-1} \mathbf{D}_{kl} \mathbf{B}_k(\xi_I, \eta_J) t_l \det(J(\xi_I, \eta_J)) W_I W_J \mathbf{u} \right) \frac{\omega_e}{\sum_{k \in N(e)} \omega_k} \quad (3.69)$$

VSO Optimization

Because of the size of the equation of the fiber orientation sensitivity it is kept simple below where one should assume that the report refers to filling in the filtered design variable inside the function in the mathematical location of the design variable. And then substituting the gradient into eq 3.64, provides:

$$\frac{dC}{d\theta_{el}} = \sum_{k \in N(e)} \frac{dC(\tilde{\theta}_{kl})}{d\tilde{\theta}_{kl}} \frac{\omega_e \rho_{el}}{\sum_{o \in N(k)} \omega_o \rho_{ol}} \quad (3.70)$$

Substituting this then into eq 3.67 provides:

$$G_2 = \frac{d\tilde{C}}{d\theta_{el}}(\rho) = \frac{\sum_{m \in N(e)} \omega_m \rho_{ml} \left(\sum_{k \in N(m)} \frac{dC(\tilde{\theta}_{kl})}{d\tilde{\theta}_{kl}} \frac{\omega_m \rho_{ml}}{\sum_{o \in N(k)} \omega_o \rho_{ol}} \right)}{\sum_{m \in N(e)} \omega_m \rho_{ml}} \quad (3.71)$$

The two filter strategies and their effect on the gradient are well applied on the model now and can be used in concurrency with the optimizer to create a roadmap. The driver itself is left and for this as explained before the GCMMA is used.

3.5. GCMMA

The solver used in this thesis is a GCMMA solver, explained in section 2.1.2. The GCMMA gives more adaptability for complex non-convex problems to gradually descend to a converged result, which is highly sought after in this thesis as the combination of the topology optimization with filters and fiber orientation optimization with filter makes this a highly non-convex problem.

3.5.1. Topology GCMMA

As mentioned before the GCMMA method uses an approximation function of the objective function to approximate the non-convex problem with a convex one. For the topology optimization objective function this becomes:

$$\tilde{C}^k(\rho) = \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} \left(\frac{p_{el}^k}{U_{el}^k - \rho_{el}} + \frac{q_{el}^k}{\rho_{el} - L_{el}^k} \right) + \gamma \quad (3.72)$$

The p_{el}^k and q_{el}^k parameters of the functions are retrieved in the GCMMA equation using the gradient computed earlier in sec 3.4.3:

$$p_{el}^k = \begin{cases} G_{1,el}(U_{el}^k - \rho_{el})^2, & G_1 \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (3.73)$$

$$q_{el}^k = \begin{cases} G_{1,el}(\rho_{el} - L_{el}^k)^2, & G_1 < 0 \\ 0, & \text{otherwise} \end{cases}$$

Where $(U_{el}^k - \rho_{el})^2$, and $(\rho_{el} - L_{el}^k)^2$ are the asymptotes scale which set how close it is allowed to move towards the edges.

To make the equation globally converge, the approximation must be bigger than the equation itself for the set domain as mentioned in section 2.1.2. The method used, is the addition of an artificial curvature $c \geq 0$ to the gradient of the optimizer which gives:

$$p_{el}^k = \begin{cases} (G_{1,el} + \frac{c_1}{\rho_{max} - \rho_{min}})(U_{el}^k - \rho_{el})^2, & G_1 \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (3.74)$$

$$q_{el}^k = \begin{cases} (G_{1,el} + \frac{c_1}{\rho_{max} - \rho_{min}})(\rho_{el} - L_{el}^k)^2, & G_1 < 0 \\ 0, & \text{otherwise} \end{cases}$$

Later to set the approximation on the same level again, one more additional term γ is added to comply with $C(\rho^k) = \tilde{C}(\rho^k)$.

The topology optimization uses a volume constraint which in GCMMA is also approximated as shown in above, this gives:

$$\tilde{V}^k(\rho) = \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} \left(\frac{p_{el}^k}{U_{el}^k - \rho_{el}} + \frac{s_{el}^k}{\rho_{el} - L_{el}^k} \right) + \beta \quad (3.75)$$

The gradient of the volume constraint function, calculated in 3.2.3, is then used in a similar manner as above in eq 3.73. This provides:

$$\begin{aligned}
r_{el}^k &= \begin{cases} (G_V + \frac{c_V}{\rho_{max} - \rho_{min}})(U_{el}^k - \rho_{el})^2, & G_V \geq 0 \\ 0, & \text{otherwise} \end{cases} \\
s_{el}^k &= \begin{cases} (G_V + \frac{c_V}{\rho_{max} - \rho_{min}})(\rho_{el} - L_{el}^k)^2, & G_V < 0 \\ 0, & \text{otherwise} \end{cases}
\end{aligned} \tag{3.76}$$

Next, the convex problem is assembled and solved using a Newton-Raphson method by stating:

$$\begin{aligned}
&\min \tilde{C}^k(\rho) \\
&\text{s.t. } \tilde{V}^k(\rho) \leq 0 \\
&\quad L_{el}^k \leq \rho \leq U_{el}^k
\end{aligned} \tag{3.77}$$

Finally, the system updates the parameters of the GCMMA solver. The asymptotes are updated based on past movements, where if 2 consistent iterations move the design variable in the same direction the asymptotes size increases, and decreases otherwise by a set factor as shown below:

$$\begin{aligned}
L_{el}^{k+1} &= \begin{cases} \rho_{el}^{k+1} - f_{incr}(\rho_{el}^k - L_{el}^k), & (\rho_{el}^{k+1} - \rho_{el}^k)(\rho_{el}^k - \rho_{el}^{k-1}) > 0 \\ \rho_{el}^{k+1} - f_{decr}(\rho_{el}^k - L_{el}^k), & \text{otherwise} \end{cases} \\
U_{el}^{k+1} &= \begin{cases} \rho_{el}^{k+1} + f_{incr}(U_{el}^k - \rho_{el}^k), & (\rho_{el}^{k+1} - \rho_{el}^k)(\rho_{el}^k - \rho_{el}^{k-1}) > 0 \\ \rho_{el}^{k+1} + f_{decr}(U_{el}^k - \rho_{el}^k), & \text{otherwise} \end{cases}
\end{aligned} \tag{3.78}$$

The artificial curvature components are also updated at this stage, they are a global mean of all the design variables inside their domain multiplied by a factor of 0.1:

$$\begin{aligned}
c_1 &= 0.1 * \frac{1}{N_{elem} N_{layers}} \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} |G_1|(\rho_{max} - \rho_{min}) \\
c_V &= 0.1 * \frac{1}{N_{elem} N_{layers}} \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} |G_V|(\rho_{max} - \rho_{min})
\end{aligned} \tag{3.79}$$

3.5.2. VSO GCMMA

The VSO optimizer follows the same steps as the topology optimizer for the GCMMA method but for the ANDFO function and gradients retrieved in 3.4.3. This results in the approximation function:

$$\tilde{C}^k(\theta) = \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} \left(\frac{p_{el}^k}{U_{el}^k - \theta_{el}} + \frac{q_{el}^k}{\theta_{el} - L_{el}^k} \right) + \gamma \tag{3.80}$$

With:

$$\begin{aligned}
p_{el}^k &= \begin{cases} (G_{2,el} + \frac{c_2}{\theta_{max} - \theta_{min}})(U_{el}^k - \theta_{el})^2, & G_2 \geq 0 \\ 0, & \text{otherwise} \end{cases} \\
q_{el}^k &= \begin{cases} (G_{2,el} + \frac{c_2}{\theta_{max} - \theta_{min}})(\theta_{el} - L_{el}^k)^2, & G_2 < 0 \\ 0, & \text{otherwise} \end{cases}
\end{aligned} \tag{3.81}$$

The curvature constraint is also added to the GCMMA solver for the fiber optimization a similar to the volume constrain as such:

$$\dot{g}^k(\theta) = \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} \left(\frac{r_{el}^k}{U_{el}^k - \theta_{el}} + \frac{s_{el}^k}{\theta_{el} - L_{el}^k} \right) + \beta \quad (3.82)$$

$$r_{el}^k = \begin{cases} (G_{\kappa,el} + \frac{c_{\kappa}}{\theta_{max} - \theta_{min}})(U_{el}^k - \theta_{el})^2, & G_{\kappa} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$$s_{el}^k = \begin{cases} (G_{\kappa,el} + \frac{c_{\kappa}}{\theta_{max} - \theta_{min}})(\theta_{el} - L_{el}^k)^2, & G_{\kappa} < 0 \\ 0, & \text{otherwise} \end{cases} \quad (3.83)$$

with the constants given as:

$$c_2 = 0.1 * \frac{1}{N_{elem} N_{layers}} \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} |G_2|(\theta_{max} - \theta_{min})$$

$$c_{\kappa} = 0.1 * \frac{1}{N_{elem} N_{layers}} \sum_{e=1}^{N_{elem}} \sum_{l=1}^{N_{layers}} |G_{\kappa}|(\theta_{max} - \theta_{min}) \quad (3.84)$$

The basic methodology of the model is now in place and all that is left is the algorithm that defines what happens at what points in the model.

3.6. Combination Method

The two optimizers are combined using a mix of concurrent and hierarchical methods to provide some feedback between both optimizers and not be primarily dependent on either one. To implement this with the current methodology an algorithm as shown below, in fig 3.6, is created. The algorithm uses 1 main outer loop which uses each GCMMA solver optimizer once concurrently and not sequentially. The GCMMA solver for each optimizer uses a small inner loop which initially retrieves the sensitivity analysis from the main outer loop and uses those gradients fixed with an iterative GCMMA adaptation to the design variables till a conservative compliance is reached or the maximum amount of iterations for that GCMMA solver is run. After the inner loop for the topology finishes the GCMMA inner loop for the fiber is run and the same system holds where either a conservative result comes out or the maximal inner iterations are run. In this way each GCMMA solver has time to find the most conservative solution and thus help avoid too fast falling into local optima.

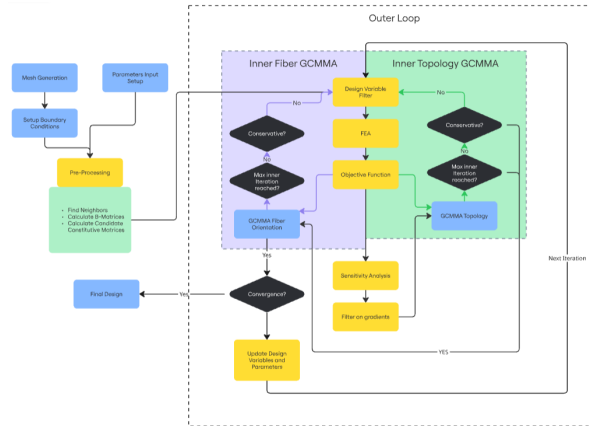


Figure 3.6: Combination Algorithm for the Laminate Optimization

After the retrieved optimized solution the model results are a bit difficult to analyze. A last post processing step can aid in explaining and analyzing some of the results obtained out of the model.

3.7. Post Processing

The AFP Machines don't use FEA fiber orientations per elements as an input but need streamlined paths for fiber steering. This final process is done in the post processing stage where the FEA optimization result is used to make a streamlined showcase of the path the machine should follow. To better show the curvature improvements, that were gotten from the optimization, a low to no smoothing method is used with simple streamline tracing over a masked vector field retrieved from the fiber orientations angles with an Euler stepper starting from seeds placed in a grid-like fashion.

First the vector field is retrieved from the fiber orientations. The first step in retrieving this is to convert each fiber angle into a vector with two scalars, the cos and sin of the double angle. The double angle is used as mentioned before in the curvature constraint section to apply a 2π period instead of a π period which is used in the angles as a fiber orientation is physically the same if it is $+45^\circ$ or $+135^\circ$, and when interpolating the vector field continuation is preferable. This thus results in:

$$\begin{aligned} v_x &= \cos(\phi) \\ v_y &= \sin(\phi) \\ \text{with } \phi &= \frac{2\pi}{N_{cand}}\theta + \pi \end{aligned} \tag{3.85}$$

This centroid vector is then interpolated over the domain of the object with a nearest interpolation where each resolution grid point has the vector for which the value is taken as the vector value of the centroid nearest him. To get the final vector field result the vectors field retrieved before is brought back to the fiber orientation angle and vectorised as such to have a correct direction field.

$$\mathbf{v}(x, y) = \begin{bmatrix} \cos(\frac{1}{2} \text{atan2}(q_y(x, y), q_x(x, y))) \\ \sin(\frac{1}{2} \text{atan2}(q_y(x, y), q_x(x, y))) \end{bmatrix} \tag{3.86}$$

With the vector field in place a density and domain mask is applied on the vector field eliminating all the vectors that have a lower density than a fixed threshold density and or elements outside of the domain. Next, a seeded grid is made for the starting locations of the streamlines inside the domain where each seed is a starting point for a streamline. These seeds are then put in an Euler stepper method where they look at the vector on that seed and take a step in the direction of the vector, if the step does not comply with a minimal curvature that is added the seed finished its run else it goes to the next pos and runs the vector Euler step again until all the streamlines are created.

Numerical Implementation

4.1. Mesh Generation

The mesh generation in this thesis was done using the gmsh python software, which is a highly efficient tool to generate any mesh for all types of geometries. It can make different mesh sizes and can be adjusted to your needs for a mesh.

The thesis uses simple gmsh scripting to create the geometries and mesh them in a way such that the element geometries form quad elements that are not too elongated so to get better and accurate results. To explain how a gmsh is scripted, the thesis follows a rectangular element with a circular hole as an example.

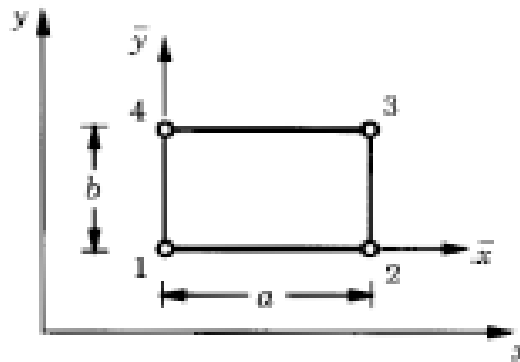


Figure 4.1: Quad element representation in FEA

The initial step of the generation is to define the geometry of the object's design space. For the example, the height H and length L of the rectangle are defined as well as the circle's radius R , and center position (c_x, c_y) . To set the reference for the mesh and optimizer later a point is also chosen to be used at the $(0, 0)$ reference point of the coordinate system as follows:

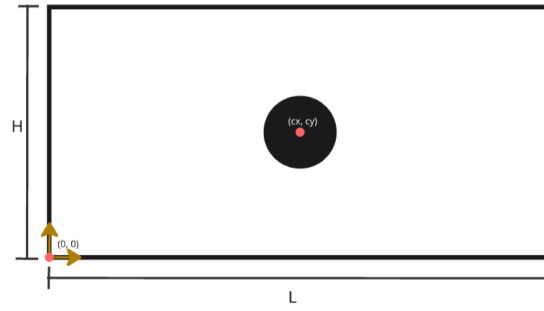


Figure 4.2: Definition of the Objects Design Space before Mesh Generation

In the next step a definition of the division of quads inside the design space is set. In this example, some symmetries are added to have more accurate results for geometrical symmetric designs. In this step the resulting section should all have 4 edges and 4 point that can be interpolated later. The circle is divided up into 8 separate arcs to maintain the symmetry and equivalently create the 4 edge point subdivisions as follows (Figure 4.3). Next, the points on the design are created

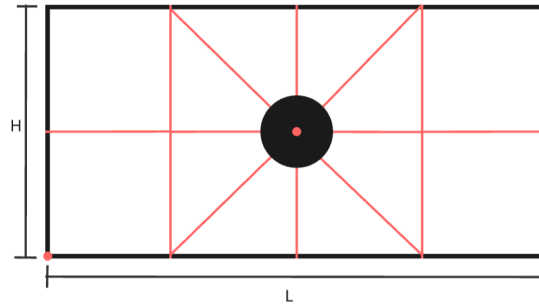


Figure 4.3: Definition division in design space before mesh generation

using the function `occ.addPoint(x, y, z)` where x and y represent the coordinates of the point inside the coordinate system set previously with the known origin. The middle section around the circle is selected to be a square in the example's case with edge length the height. In the examples these points are:

Table 4.1: Coordinates of mesh points

Point	x	y	Point	x	y
1	0.00	0.00	12	$(L+H)/2$	$H/2$
2	$(L-H)/2$	0.00	13	$L/2$	0.00
3	$(L-H)/2$	$H/2$	14	$L/2$	H
4	$(L-H)/2$	H	15	$L/2+R$	$H/2$
5	0.00	H	16	$L/2 + R \cos(45)$	$H/2 + R \cos(45)$
6	0.00	$H/2$	17	$L/2$	$H/2 + R$
7	$(L+H)/2$	0.00	18	$L/2 - R \cos(45)$	$H/2 + R \cos(45)$
8	L	0.00	19	$L/2 - R$	$H/2$
9	L	$H/2$	20	$L/2 - R \cos(45)$	$H/2 - R \cos(45)$
10	L	H	21	$L/2$	$H/2 - R$
11	$(L+H)/2$	H	22	$L/2 + R \cos(45)$	$H/2 - R \cos(45)$

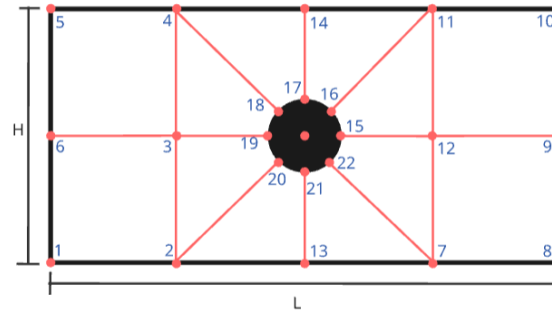


Figure 4.4: Location of the points set for mesh generation

After the points are created the line edges are formed, two different functions are used, for straight lines the gmsh script uses `occ.addLine(Point1, Point2)` where a line is drawn from *Point1* to *Point2*. For arc lines, a different function is used `occ.addCircleArc(Point1, centerPoint, Point2)` where once again a line is drawn from *Point1* to *Point2* but in an arc with center point the *centerPoint* variable. It is very important to keep the direction of the lines known. The list of lines for the example are shown below:

Table 4.2: Mesh Lines

Line	<i>Point1</i>	<i>Point1</i>	Line	<i>Point1</i>	<i>Point2</i>
1	1	2	18	14	4
2	2	3	19	2	20
3	3	4	20	13	21
4	4	5	21	7	22
5	5	6	22	12	15
6	6	1	23	11	16
7	6	3	24	14	17
8	7	8	25	4	18
9	8	9	26	3	19
10	9	10	27	15	16
11	10	11	28	16	17
12	11	12	29	17	18
13	12	7	30	18	19
14	12	9	31	19	20
15	2	13	32	20	21
16	13	7	33	21	22
17	11	14	34	22	15

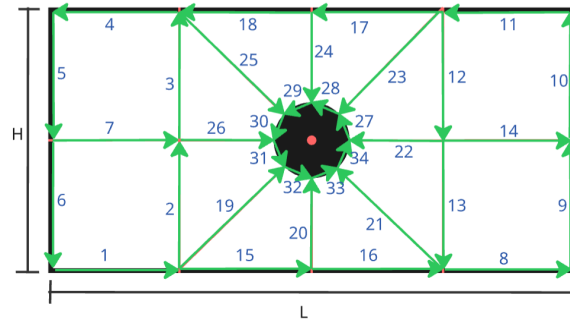


Figure 4.5: Line direction setup for the mesh generation

Next the surfaces are created using the previous created lines. To create the surface a double function can be used, `occ.addPlaneSurface([occ.addCurveLoop([Line1,Line2,Line3,Line4]))]`. To create these surfaces correctly it is important to follow the directions of the lines and to make sure that the point between the lines are the same point objects. If a line direction is in reverse then the line can be written with a minus sign to reverse its direction. To make sure later that the mesh is correctly generated later and that the sections can fit together, it is important to make sure that all surfaces follow a clockwise direction. In the example the surfaces are represented as follows:

Table 4.3: Mesh Surfaces

Surface	<i>Line1</i>	<i>Line2</i>	<i>Line3</i>	<i>Line4</i>
1	1	2	-7	6
2	7	3	4	5
3	8	9	-14	13
4	14	10	11	12
5	15	20	-32	-19
6	19	-31	-26	-2
7	26	-30	-25	-3
8	25	-29	-24	18
9	24	-28	-23	17
10	23	-27	-22	-12
11	22	-34	-21	-13
12	21	-33	-20	16

Finally, the surfaces are synchronized together so that the mesh is resultant into 1 single object. After the synchronization, the discretizations of the object into small element is done. The gmsh does this by dividing each edge into discrete lengths where nodes are placed on each segment alongside an edge of a surface. The function `gmsh.model.mesh.setTransfiniteCurve(Line1, numberOfNodes)` is used where the *numberOfNodes* is the amount of nodes that are set alongside the line *Line1*. In this example the lines that go from the edges towards the circle are also progressively segmented, by placing nodes bit by bit closer to each other the closer you get to the circle. The surfaces are also discretized using the `gmsh.model.mesh.setTransfiniteSurface(surface1)` function, where *surface1* is discretized into triangles using the nodes or points discretized from the edges of that surface. They are then recombined into quad elements using the `gmsh.model.mesh.setRecombine(2, surface1)` function and after this the mesh is generated and the result for this example is shown in Figure 4.6.

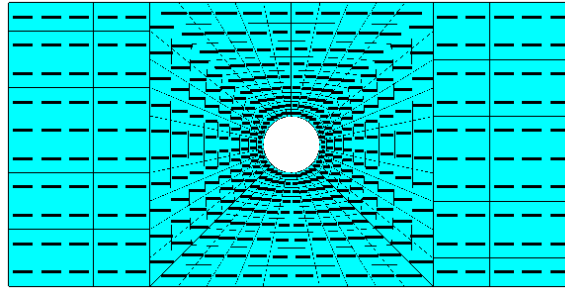


Figure 4.6: The mesh result from the example

After the mesh is generated using gmsh it is saved as a file in the software and can then be loaded in to use inside the optimizer. The elements and nodes are extracted from the gmsh file and only the useful nodes are selected to pass onto the optimizer as helping points, like the center point of the circle, is also saved as a node inside the gmsh file and not necessarily for the optimizer.

4.2. Object and Run Setup

In this section the program get around is explained to set up the mesh and boundary conditions and how to go about setting certain parameters or viewing certain properties.

4.2.1. Boundary Conditions

After the mesh has been generated, the boundary conditions to the mesh can be added, as mentioned in the methodology the boundary conditions that can be added consist of edge traction, node forces, or displacements of nodes or edges. The GUI in the software has the following look:

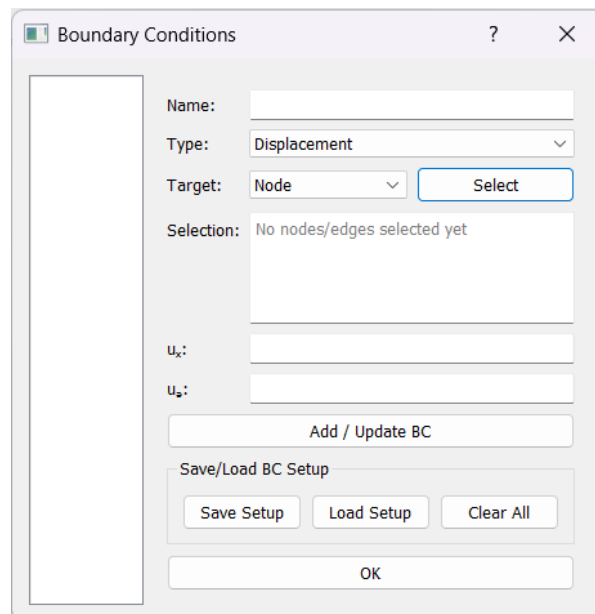


Figure 4.7: Boundary Conditions UI

To make a boundary, the user of the software has to give the boundary condition set a name. The type of boundary condition can be either loading or displacement after which you will also see the variables below change accordingly. Also the target of the boundary condition can be chosen between either a boundary condition on a node or edge, after which the edge or node can be selected. To select the edge or node the user can shift click drag a rectangle over the edges or nodes they want to select and the nodes or edges will be shown in red, to confirm selection the user can press enter and the nodes

or pair of nodes for edges will show up in the menu screen and to finally confirm the user presses the add button after which the BCs, are uploaded the BCs calculator object where they are processed to make the force and displacement vector eventually.

The processing of the BCs is done inside file BCsTranslation.py where it separates the edges and nodes at which the boundaries are applied and if it is a nodal displacement or force it is directly added to the BCs lists and force and displacement vector. However, in the case of a loading on an edge a Gauss integration is firstly don't to retrieve the resulting forces on the node of the edge which is the passes on to the vectors.

The user can also save and or load from a save file the boundary conditions setups he creates as to not lose time having to recreate the same boundary conditions if other tests want to be run. When the user then finishes he exits the menu by pressing ok and can continue to the parameters page to set up the parameters off the simulation.

4.2.2. Parameters

There is a high quantity of parameters that the user can play with to set up the system of the optimization of the composite object. A list of all the parameters and their description can be found below:

Table 4.4: Model parameters and descriptions

Parameter	Type	Description
N_{cand}	int	Number of candidate fiber orientations
V_f	float	Volume fraction representing the material resource constraint
N_{layers}	int	Number of laminate layers
x_{init}	float	Initial material density value
y_{init}	float	Initial fiber orientation value
γ	float	Initial density penalization parameter
$\dot{\gamma}$	float	Update rate of density penalization parameter
γ_{max}	float	Maximum allowable density penalization
$p_{\sigma,init}$	float	Initial fiber penalty parameter
k_{max}	int	Number of iterations for constant $p_{\sigma,init}$ stage
α	float	Scale for the $p_{\sigma,init}$ linear decreasing stage
x_{th}	float	Density threshold for the $p_{\sigma,init}$ adaptive stage
w_{th}	float	Weight threshold for the $p_{\sigma,init}$ adaptive stage
a	float	Coefficient A for penalty function in the $p_{\sigma,init}$ adaptive stage
b	float	Coefficient B for penalty function in the $p_{\sigma,init}$ adaptive stage
c	float	Coefficient C for penalty function in the $p_{\sigma,init}$ adaptive stage
d	float	Coefficient D for penalty function in the $p_{\sigma,init}$ adaptive stage
ϵ	float	Convergence level before activating the $p_{\sigma,init}$ adaptive stage
r_x	float	Filter radius for density design variable

Continued on next page

Table 4.4 – continued from previous page

Parameter	Type	Description
r_y	float	Filter radius for fiber orientation variable
$r_{y,p}$	float	Power parameter for fiber filter
$asym_{l,x}$	float	Lower asymptote for density update
$asym_{u,x}$	float	Upper asymptote for density update
$move_x$	float	Move limit for density variable
$asym_{l,y}$	float	Lower asymptote for fiber orientation update
$asym_{u,y}$	float	Upper asymptote for fiber orientation update
$move_y$	float	Move limit for fiber orientation variable
topo_opt	bool	Flag to enable topology optimization
fiber_opt	bool	Flag to enable fiber orientation optimization
topo_filter	bool	Flag to enable topology design variable filtering
fiber_filter	bool	Flag to enable fiber design variable filtering
fiber_sens_filter	bool	Flag to enable fiber sensitivity filtering
runtype	str	Optimization run selection
$Iter_{max}$	int	Maximum number of global optimization iterations
E_1	float	Longitudinal modulus of elasticity (MPa)
E_2	float	Transverse modulus of elasticity (MPa)
G_{12}	float	In-plane shear modulus (MPa)
G_{23}	float	Transverse shear modulus (MPa)
ν_{12}	float	Poisson's ratio
ρ	float	Material density (kg/m ³)

The parameters are set up in the parameter page of the program where everything can be set and selected to try out different parameters, shown below in fig 4.8. A parameter setup can also be saved and loaded in if previous parameters were used to facilitate the re-testing of similar setups.

Figure 4.8: Parameter page of the program

4.2.3. Properties views

After everything has been prepped for the run, a set of properties can be selected in the Run window. The user can select the properties that will show up during the run. They are during the run every x amount of iterations being updated in each of their respective mesh view with hue colors and density saturation of set in the code. The user can also select which specific layer is desired to show these properties if more are selected. The properties that are selectable and their description are shown below:

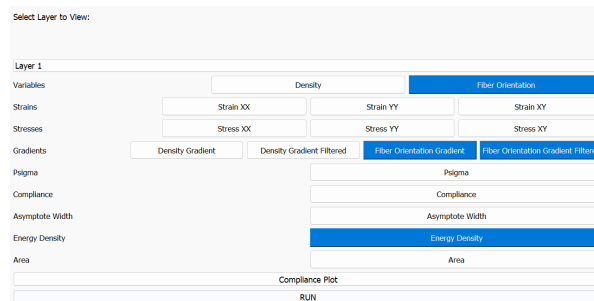


Figure 4.9: Runpage with property selection to view

Table 4.5: Selectable property views and descriptions

Property	Unit	Description
Variables		
Density	(—)	Element-wise material density distribution
Fiber Orientation	(—) or deg	Element-wise fiber orientation variable
Strains		
Strain XX	(—)	Normal strain in the x -direction (ε_{xx})
Strain YY	(—)	Normal strain in the y -direction (ε_{yy})
Strain XY	(—)	Shear strain in the xy -plane (ε_{xy})
Stresses		
Stress XX	MPa	Normal stress in the x -direction (σ_{xx})
Stress YY	MPa	Normal stress in the y -direction (σ_{yy})
Stress XY	MPa	Shear stress in the xy -plane (σ_{xy})
Gradients		
Density Gradient	(—)	Density sensitivity of the objective
Density Gradient Filtered	(—)	Filtered density sensitivity after spatial filtering
Fiber Orientation Gradient	(—)	Fiber sensitivity of the objective
Fiber Orientation Gradient Filtered	(—)	Filtered fiber orientation sensitivity
Global / Derived Quantities		
Psigma	(—)	Current fiber penalty parameter value
Compliance	N·mm or J	Structural compliance (objective function value)
Asymptote Width	(—)	Width between upper and lower asymptotes
Energy Density	J/m ³	Strain energy per unit volume
Area	m ²	Total material area of the current design

On top of the viewable properties of the mesh, the Layer view page and the Mesh view page are updated. The Layer view page shows the density and fiber orientation throughout the whole mesh for each layer separately, and thus has the best representation of all the demanded design variables. The Mesh view page on the other hand is a mean of all the densities of the elements in each layer and can thus be represented as the overall view of the density in the design space. In this page the user can also check the mesh displacement due to the applied boundary conditions with the current object and can be scaled using the slide bar.

4.3. The process

The process section is the key optimization section of the numerical integration. It's divided into 3 sections which are executed in order inside the program. The pre-processing which deals with all pre-computations to make the iterations more efficient, the optimization itself where the algorithm of each iteration is explained and the calculations with the functions that are applied, and the post-processing where the fiber orientations are used to create a streamline map of the results.

4.4. Pre-Processing

The pre-processing is set up to eliminate double calculations during the iterations where they are not necessary. These calculations include the calculation of candidate angles from their numbers, the constitutive ply matrices corresponding to those angles. The resource and area calculations of all elements for this specific design space. The centroids and neighbor pre-computations for the filtering later. The constant volume gradient used in the topology optimization GCMMA, and finally, the shape functions, jacobians and B matrices for all the elements.

`_calculate_candidate_angles()`

This function has as input the number of candidates selected in the parameter page. It uses a simple formula to translate the number of candidates N_{cand} into a well divided range of candidate angles *CandidateAngles* ranging over all possible angles for the ply.

$$CandidateAngles = [\frac{180}{i}(i+1) - 90 \quad for \quad i = [1, N_{cand}]] \quad (4.1)$$

Table 4.6: Calculate Candidate constitutive matrix

Inputs	desc
N_{Cand}	the number of candidates
Outputs	desc
<i>CandidateAngles</i>	array with candidate angles

`_calculate_candidate_elastic_constitutive_matrix()`

This function creates the constitutive matrices corresponding to each candidate angle retrieved from the previous function and the material properties set in the parameters page. The equations applied are shown in the methodology section using equation 3.7.

`_calculate_element_centroids()`

In this function the centroids of the quad elements are retrieved from the coordinated of the node of that element. The input into this function is an array of the nodal coordinates corresponding to each element. And the mean of all 4 nodal coordinates is calculated.

`_calculate_element_areas()`

The element areas can be retrieved similarly as the centroids from the nodal coordinates of the elements. The following functions hold for the area of the element's coordinates using the shoelace formula:

Table 4.7: Calculate Constitutive matrices of candidate angles

Inputs	desc
<i>CandidateAngles</i>	array with candidate angles
<i>MaterialProperties</i>	List with all the corresponding material properties
Outputs	desc
<i>Q</i>	Constitutive matrices of candidate angles as np.array of size $(N_{cand}, 3, 3)$

Table 4.8: Calculate Element Centroids

Inputs	desc
<i>nodecoord</i>	array with the nodal coordinates of each element of size $(N_{elem}, 4, 2)$
Outputs	desc
<i>centroids</i>	array with all the central point of each element of size (N_{elem})

$$A^{(e)} = \frac{1}{2} \left| \sum_{i=1}^4 \left(x_i^{(e)} y_{i+1}^{(e)} - y_i^{(e)} x_{i+1}^{(e)} \right) \right|, \quad x_5^{(e)} = x_1^{(e)} \quad (4.2)$$

Table 4.9: Calculate Element Areas

Inputs	desc
<i>nodecoord</i>	array with the nodal coordinates of each element of size $(N_{elem}, 4, 2)$
Outputs	desc
<i>areas</i>	array with all the central point of each element of size (N_{elem})

`_precompute_neighbors()`

In this functions the centroids of the elements *centroids* retrieved from the previous corresponding function, and the parameters of the fiber optimization radius R_ρ , and the density optimization radius R_θ retrieved from the parameter section before. The method makes a (N_{elem}, N_{elem}) matrix by subtracting the centroid coordinates by each other in a transposed form and computing the distance of the resulting vectors. The output is a diagonal symmetric matrix where the combination of 2 elements had the value of it's distance between them inside. Then a filter of any value bigger than the radii mentioned before is placed on this matrix to retrieve a matrix of 1's and 0's representing elements which are each other neighbors within those radii.

`_precompute_neighbor_differences_and_weights()`

In this function the same method as before for the neighbors differences is computed and the weights from these computations are calculated using equation 3.58 and equation 3.63 from the methodology. After the weight calculation a filter is applied by multiplying by the neighbors 1/0 matrix obtained in the previous functions to only leave the functional weights of the neighbors.

`_precompute_derivative_shape_functions()`

This function precomputes the derivatives of the shape functions for the GaussPoints and the center of the quad elements in local coordinates using equation 3.15.

Table 4.10: Find Element Neighbors

Inputs	desc
$centroids$	array with all the central point of each element of size (N_{elem})
R_ρ	filter radius of the density design variable
R_θ	filter radius of the fiber orientation
Outputs	desc
$\rho_{neighbors}$	matrix of size (N_{elem}, N_{elem}) representing the neighbors of all the elements of the density
$\theta_{neighbors}$	matrix of size (N_{elem}, N_{elem}) representing the neighbors of all the elements of the fiber orientation

Table 4.11: Calculate Element Neighbors weights

Inputs	desc
$centroids$	array with all the central point of each element of size (N_{elem})
R_ρ	filter radius of the density design variable
R_θ	filter radius of the fiber orientation
$\rho_{neighbors}$	matrix of size (N_{elem}, N_{elem}) representing the neighbors of all the elements of the density
$\theta_{neighbors}$	matrix representing the neighbors of all the elements of the fiber orientation
$p_{R\theta}$	the power of the filter radius weight of the fiber orientation
Outputs	desc
ω_ρ	matrix of size (N_{elem}, N_{elem}) representing the weight of all the elements neighbors of the density
ω_θ	matrix of size (N_{elem}, N_{elem}) representing the weights of all the elements neighbors of the fiber orientation

Table 4.12: Precompute derivative shape functions

Inputs	desc
$GaussPoints$	The GaussPoints coordinates of the local system.
$Center$	the gauss representation of the center point of the element being (0,0)
Outputs	desc
$\frac{dN}{d\xi}$	Array of the derivative shape function values for all the 4 nodal GaussPoints
$\frac{dN}{d\eta}$	Array of the derivative shape function values for all the 4 nodal GaussPoints
$\frac{dN}{d\xi}(center)$	Array of the derivative shape function values for the center
$\frac{dN}{d\eta}(center)$	Array of the derivative shape function values for the center

`set_jacobians()`

This function calculates the jacobians, inverse Jacobians, and the determinants of the jacobians for all the elements. It uses the element's nodal coordinates and shape functions derivatives following equation 3.16 and equation 3.17.

Table 4.13: Precompute Jacobians and related matrices

Inputs	desc
<i>nodecoord</i>	array with the nodal coordinates of each element of size $(N_{elem}, 4, 2)$
$\frac{dN}{d\xi}$	Array of the derivative shape function values for all the 4 nodal GaussPoints
$\frac{dN}{d\eta}$	Array of the derivative shape function values for all the 4 nodal GaussPoints
$\frac{dN}{d\xi}(center)$	Array of the derivative shape function values for the center
$\frac{dN}{d\eta}(center)$	Array of the derivative shape function values for the center
Outputs	desc
<i>GaussJacobians</i>	Array of size $(N_{elem}, 4, 2, 2)$, representing the Jacobians for each node's GaussPoint of each element
<i>GaussJacobiansCentered</i>	Array of size $(N_{elem}, 2, 2)$, representing the Jacobians for the center of each element
<i>GaussDeterminant</i>	Array of size $(N_{elem}, 4)$, representing the Jacobian determinants for each node's GaussPoint of each element
<i>GaussDeterminantCentered</i>	Array of size (N_{elem}) , representing the Jacobian determinant for the center of each element
<i>GaussJacobiansInv</i>	Array of size $(N_{elem}, 4, 2, 2)$, representing the Inverse of the Jacobians for each node's GaussPoint of each element
<i>GaussJacobiansInvCentered</i>	Array of size $(N_{elem}, 2, 2)$, representing the Inverse of the Jacobians for the center of each element

`set_B_matrices()`

This function calculates the B Matrix for each element from the previously calculated jacobian information and the derivative shape functions. The equations applied can be found in the methodology (see equation 3.18). The B matrices are made for all the 4 GaussPoints and the CenterPoint.

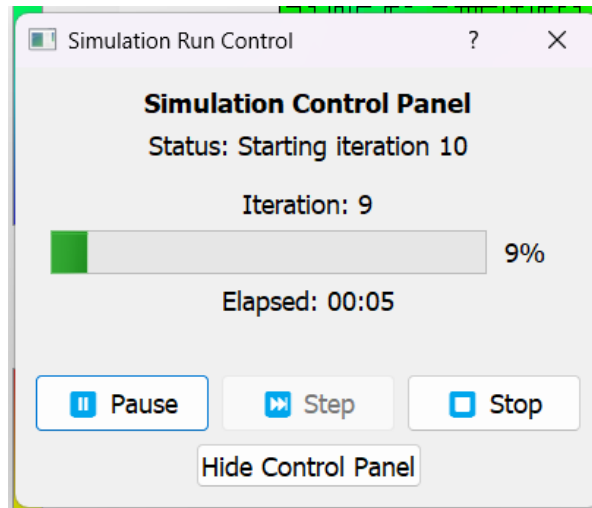
As mentioned before these preprocessed properties and values are calculated before the iteration loop to make the iterations more efficient and less computationally expensive. Once the preprocessing is finished the algorithm starts its first iteration.

4.5. Optimization

During the initialization of the optimization itself there are some prepped features of the program starting up, the view shift from the runpage to the property page, so the user can directly watch the change in the properties during the run. The control panel for the run, show in fig 4.10, is also initialized. The control panel is a module the user can use to pause the iteration loop at certain points if the user wishes to analyze certain properties or aspect off the mesh or elements during a certain part of the run. The user Can also stop the run to retry with different parameters and rerun the optimizer.

Table 4.14: Precompute Jacobians and related matrices

Inputs	desc
<i>GaussJacobiansInv</i>	Array of size $(N_{elem}, 4, 2, 2)$, representing the Inverse of the Jacobians for each node's GaussPoint of each element
<i>GaussJacobiansInvCentered</i>	Array of size $(N_{elem}, 2, 2)$, representing the Inverse of the Jacobians for the center of each element
$\frac{dN}{d\xi}$	Array of the derivative shape function values for all the 4 nodal GaussPoints
$\frac{dN}{d\eta}$	Array of the derivative shape function values for all the 4 nodal GaussPoints
$\frac{dN}{d\xi}(center)$	Array of the derivative shape function values for the center
$\frac{dN}{d\eta}(center)$	Array of the derivative shape function values for the center
Outputs	desc
<i>GaussBMatrices</i>	Array of size $(N_{elem}, 4, 3, 8)$, representing the B matrices of each element's GaussPoints
<i>CenterBMatrices</i>	Array of size $(N_{elem}, 3, 8)$, representing the B matrices of each element's center point

**Figure 4.10:** Control Panel of the Simulation Run

The iteration is then started and counted in a loop. This loop, shown in figure 4.10, has different sections to retrieve the necessary properties and other optimization variables, as well as the GCMMA solvers at the end of each iteration.

```
_filter_x()
```

If the user has activated the design variable filtering then this is the first function ran by the loop. It filter the density design variable on its neighbors using the previously pre-calculated neighbors matrix and weights. The current density is used as an input into equation 3.57, and the filtered density comes out as an output array of the same size.

```
_filter_fibers()
```

After the filtering of the density design variable, the fiber orientation's design variable is also filtered

Table 4.15: Filter of Density Design Variable

Inputs	desc
ρ	the density design variable of size (N_{elem}, N_{layers})
$\rho_{neighbors}$	matrix of size (N_{elem}, N_{elem}) representing the neighbors of all the elements of the density
ω_ρ	matrix of size (N_{elem}, N_{elem}) representing the weight of all the elements neighbors of the density
Outputs	desc
$\tilde{\rho}$	the filtered density design variable of size (N_{elem}, N_{layers})

using equation 3.62 (with weights from equation 3.63), if activated in the parameters page beforehand. Depending on whether the density filter was applied the filtered or normal density DV is inputted into this filtering.

Table 4.16: Filter of Fiber Design Variable

Inputs	desc
θ	the fiber design variable of size (N_{elem}, N_{layers})
ρ or $\tilde{\rho}$	the inputted density design variable of size (N_{elem}, N_{layers})
$\theta_{neighbors}$	matrix of size (N_{elem}, N_{elem}) representing the neighbors of all the elements of the fiber
ω_θ	matrix of size (N_{elem}, N_{elem}) representing the weights of all the elements neighbors of the fiber orientation
Outputs	desc
$\tilde{\theta}$	the filtered fiber design variable of size (N_{elem}, N_{layers})

`_calculate_NDFO_interpolated_constitutive_matrices()`

In this function the weight and equivocally the constitutive matrices for each ply from the NDFO method are calculated from the preprocessed constitutive matrices of the candidate angles, the current either filtered or non-filtered fiber orientation design variables and the adaptive $P\sigma$'s, using equation 3.45.

Table 4.17: Constitutive matrix calculation of the fiber orientations

Inputs	desc
$P\sigma$	the normal deviation of the distributions used to calculate the weight of the fiber orientations as an array of size (N_{elem}, N_{layers})
Q	Constitutive matrices of candidate angles as np.array of size $(N_{cand}, 3, 3)$
θ or $\tilde{\theta}$	the inputted fiber design variable of size (N_{elem}, N_{layers})
Outputs	desc
Q^{NDFO}	an array with, the weighted constitutive matrix from the fiber orientations, of size $(N_{elem}, N_{layers}, 3, 3)$

`_apply_SIMP_to_constitutive_matrices()`

Now that the constitutive matrices have been interpolated from the weight of the candidate angles with reference to the fiber orientation design variable, the application of the density follows. In this application

the SIMP method as mentioned in the methodology is used to apply a fraction on the constitutive matrix corresponding to it's equivalent density representation. The equation used can be found in the methodology section (see equation 3.30).

Table 4.18: SIMP Method application on the constitutive matrices

Inputs	desc
Q^{NDFO}	an array with, the weighted constitutive matrix from the fiber orientations, of size $(N_{elem}, N_{layers}, 3, 3)$
ρ or $\tilde{\rho}$	the inputted density design variable of size (N_{elem}, N_{layers})
Outputs	desc
Q^{SIMP}	an array with, the SIMP applied constitutive matrices, of size $(N_{elem}, N_{layers}, 3, 3)$

`_calculate_ABD_matrices()`

Using the constitutive matrices retrieved from the simp application, the ABD matrices of all the element are calculated by using the integration of each layer represented by a corresponding constitutive matrix over the thickness of that layer and added together. The formula used for this is equation 3.1.

Table 4.19: Calculation of the ABD matrices from the constitutive matrices

Inputs	desc
Q^{SIMP}	an array with, the SIMP applied constitutive matrices, of size $(N_{elem}, N_{layers}, 3, 3)$
$layerThickness$	The thickness of any layer in mm
N_{layers}	The amount of layer in the current design domain.
Outputs	desc
ABD	an array with, the ABD matrices for each element, of size $(N_{elem}, 6, 6)$

`_calculate_element_stiffness_matrices_D_matrix()`

The next step in the iteration is to retrieve the element's stiffness matrices. The equations retrieved in the methodology section (see equation 3.20 and equation 3.22) are applied with the Q matrices from the SIMP application function. Together in combination with the precalculated B matrix, Gauss determinants and Weights the element stiffness matrix is found.

`_assemble_global_stiffness_matrix()`

After retrieving the element's stiffness matrices they are assembled in a global stiffness matrix using a vectorized sparse matrix assembly which is a high computational method to assemble the stiffness matrices together in their corresponding DOF map. The matrix is then transformed into a CSR (Compressed Sparse Row) format, where only the non-zero values are saved with their corresponding indexes for fast linear solving.

`_solve_for_displacements()`

Now with the Boundary conditions that were applied in the previous section, and the now calculated global stiffness matrix, the resulting displacement matrix of the current iterations can be calculated by the linear matrix solver. Before the solver the matrices are first partitioned into their *FixedNodes* and *FreeNodes* retrieved from the BCs displacements applied on the domain. The corresponding partitioned global stiffness, force, and known displacement matrix are then used to calculate the remaining unknown displacement (see equation 3.27), and then the displacement matrix is reassembled into 1 matrix.

Table 4.20: Calculation of the element's stiffness matrix

Inputs	desc
Q^{SIMP}	an array with, the SIMP applied constitutive matrices, of size $(N_{elem}, N_{layers}, 3, 3)$
$layerThickness$	The thickness of any layer in mm
$GaussBMatrices$	Array of size $(N_{elem}, 4, 3, 8)$, representing the B matrices of each element's Gauss Points
$GaussWeights$	Array of size (4), representing the weights of each Gauss Point
$GaussDeterminant$	Array of size $(N_{elem}, 4)$, representing the Jacobian determinants for each node's Gauss Point of each element
Outputs	desc
K_e	an array with the element stiffness matrices for each element, of size $(N_{elem}, 8, 8)$

Table 4.21: Calculation of the global stiffness matrix

Inputs	desc
K_e	an array with the element stiffness matrices for each element, of size $(N_{elem}, 8, 8)$
$elementDOF$	an array with the element's degree of freedom index, of size $(N_{elem}, 8)$
Outputs	desc
K_G	a singular matrix representing the global stiffness of the object for all it's DOF in CSR form for faster computation later

Table 4.22: Calculation of the global stiffness matrix

Inputs	desc
K_G	a singular matrix representing the global stiffness of the object for all it's DOF in CSR form for faster computation later
F	the force matrix representing the applied forces on the object.
U	the displacement matrix filled with the known fixed displacements
$FixedNodes$	an array with the DOFs of the fixed nodes
$FreeNodes$	an array with the DOFs of the free nodes
Outputs	desc
U	the displacement matrix now representing the whole displacement matrix of all nodes.

`_calculate_element_strains()`

Now using the known behaviour of the elements inside the design domain, the strains can be calculated i correspondence between each element's center B matrix and nodal displacements. The equation used for this is the linear strain–displacement equation in equation 3.12. The resulting strain matrix can be subdivided into the axial strains and the engineering shear strain of the elements. The center point of the element is used as a representation of the strain on average in that element however any strain in any part of the element can be calculated using it's corresponding B matrix.

Table 4.23: Calculation of the strain matrices

Inputs	desc
<i>GaussBMatricesCentered</i>	Array of size $(N_{elem}, 3, 8)$, representing the B matrices of each element's Center Gauss Points
<i>elementDOF</i>	an array with the element's degree of freedom index, of size $(N_{elem}, 8)$
<i>U</i>	the displacement matrix now representing the whole displacement matrix of all nodes.
Outputs	desc
ϵ	the strain matrix representing the axial and shear strains in each element's center point, of size $(N_{elem}, 3)$.

`_calculate_element_stresses()`

Similar to how the strains are calculated, the stresses are also calculated at the elements center from the previously calculated strains. These calculations are done through the stress-strain relation in equation 3.2.

Table 4.24: Calculation of the stresses matrices

Inputs	desc
<i>ABD</i>	an array with, the ABD matrices for each element, of size $(N_{elem}, 6, 6)$
ϵ	the strain matrix representing the axial and shear strains in each element's center point, of size $(N_{elem}, 3)$.
Outputs	desc
σ	the stress matrix representing the axial and shear stresses in each element's center point, of size $(N_{elem}, 3)$.

`_calculate_element_strain_energy()`

The strain energies can be derived from the displacement behavior of the object and its stiffness matrices using the elastic strain energy / compliance relation in equation 3.29.

`_calculate_global_compliance_from_energy()`

The global compliance calculated is easily derivable from its relation to the strain energy as shown before is the compliance is simply the double of the strain energy. To then retrieve the total compliance of the whole object, all the elements compliance are summed.

`_calculate_gradient_x_element_stiff()`

The following functions correspond to the sensitivity calculation shown in the methodology section 3.2.2. The first function is the calculation of the gradient for the density design variable, where the global

Table 4.25: Calculation of the strain energy matrices

Inputs	desc
u	the displacement matrix now representing the whole displacement matrix of all nodes.
K_e	an array with the element stiffness matrices for each element, of size $(N_{elem}, 8, 8)$
$elementDOF$	an array with the element's degree of freedom index, of size $(N_{elem}, 8)$
Outputs	desc
U	the strain energy of each element at the current iteration, an array of size (N_{elem})

Table 4.26: Calculation of the global compliance matrix

Inputs	desc
U	the strain energy of each element at the current iteration, an array of size (N_{elem})
Outputs	desc
C	the total compliance of the object inside the design space

displacement vector and the stiffness matrices can be used to simplify the computation as shown in equation 3.42 by simply multiplying it by the power and dividing over the current density design variable or its filtered version if applicable.

Table 4.27: Calculation of the density gradient matrix

Inputs	desc
p_ρ	The density penalty power value
ρ or $\tilde{\rho}$	the inputted density design variable of size (N_{elem}, N_{layers})
u	the displacement matrix now representing the whole displacement matrix of all nodes.
K_e	an array with the element stiffness matrices for each element, of size $(N_{elem}, 8, 8)$
$elementDOF$	an array with the element's degree of freedom index, of size $(N_{elem}, 8)$
Outputs	desc
$\frac{dC}{d\rho}$	the gradient matrix of the density design variables, of size (N_{elem}, N_{layers})

`_filter_gradient_x()`

The next sensitivity function follows the filtered density variable application for the sensitivity of the design variable. The equation in equation 3.59 uses as mentioned before the simple method of multiplying the previous calculated gradient by a weight fraction of the neighbors.

`_calculate_gradient_y()`

The fiber orientation design variable follows if applicable a three step functions method to calculate the gradients, apply the filters off the design variables and then another filter for the sensitivity itself. The first function for the sensitivity of the fiber orientation design variable itself follows Eq. 3.47.

Table 4.28: Calculation of the density filtered gradient matrix

Inputs	desc
$\frac{dC}{d\rho}$	the gradient matrix of the density design variables, of size (N_{elem}, N_{layers})
ω_ρ	matrix of size (N_{elem}, N_{elem}) representing the weight of all the elements neighbors of the density
Outputs	desc
$\tilde{\frac{dC}{d\rho}}$	the gradient matrix of the density design variables, of size (N_{elem}, N_{layers})

Table 4.29: Calculation of the fiber orientation gradient matrix

Inputs	desc
N_{cand}	number of candidate fiber angles (scalar)
θ	fiber orientation design variable (or filtered variant), of size (N_{elem}, N_{layers})
P^σ	normal deviation parameter for the ANDFO weights, of size (N_{elem}, N_{layers})
$\mathbf{Q}$	constitutive matrices of the candidate angles, of size $(N_{cand}, 3, 3)$
$\tilde{\rho}$	filtered density design variable used for SIMP scaling, of size (N_{elem}, N_{layers})
p_ρ	density penalty power value used in SIMP (scalar)
t_l	layer thickness in mm (scalar)
$\mathbf{B}$	Gauss point B-matrices, of size $(N_{elem}, N_{gauss}, 3, 8)$
$\det(\mathbf{J})$	Gauss point Jacobian determinants, of size (N_{elem}, N_{gauss})
$\mathbf{W}$	Gauss weights, of size (N_{gauss})
U	global displacement vector of the current iteration (size N_{dof})
$elementDOF$	element DOF map, of size $(N_{elem}, 8)$
Outputs	desc
$\frac{dC}{d\theta}$	gradient matrix of the fiber orientation design variables, of size (N_{elem}, N_{layers})

`_filter_gradient_y_density()`

The next function of the gradient calculation of the fiber orientation design variable is used if the fiber filter is applicable. It uses a similar equation as the one used for the topology optimization but with the addition of the density design variable as a parameter. And dependent on whether a density filter is applied this also determines whether the filtered or non-filtered design variable is used.

Table 4.30: Calculation of the fiber orientation gradient matrix with filtered orientation

Inputs	desc
$\frac{\tilde{d}C}{d\theta}$	the gradient matrix of the fiber orientation design variables, of size (N_{elem}, N_{layers})
ω_θ	matrix of size (N_{elem}, N_{elem}) representing the weights of all the elements neighbors of the fiber orientation
ρ or $\tilde{\rho}$	the inputted density design variable of size (N_{elem}, N_{layers})
Outputs	desc
$\frac{\tilde{d}C}{d\theta}$	the gradient matrix of the fiber orientation design variables, of size (N_{elem}, N_{layers})

`_filter_gradient_y_sensitivity()`

The final function of the sensitivity section is the additional filtering of the sensitivity of the fiber orientation with a sensitivity filter, as shown in equation 3.67

Table 4.31: Calculation of the fiber orientation gradient matrix

Inputs	desc
$\frac{\tilde{d}C}{d\theta}$	the gradient matrix of the fiber orientation design variables, of size (N_{elem}, N_{layers})
ρ or $\tilde{\rho}$	the inputted density design variable of size (N_{elem}, N_{layers})
ω_θ	matrix of size (N_{elem}, N_{elem}) representing the weights of all the elements neighbors of the fiber orientation
Outputs	desc
$\frac{\tilde{d}C}{d\theta}$	the gradient matrix of the fiber orientation design variables, of size (N_{elem}, N_{layers}) with applied sensitivity filter.

4.6. Post-Processing

After the optimization of the problem a post processing step can be applied on the result to retrieve the fiber streamline paths out of the result. This is done in the last page of the main GUI named Fiber Path. A window will open as such:

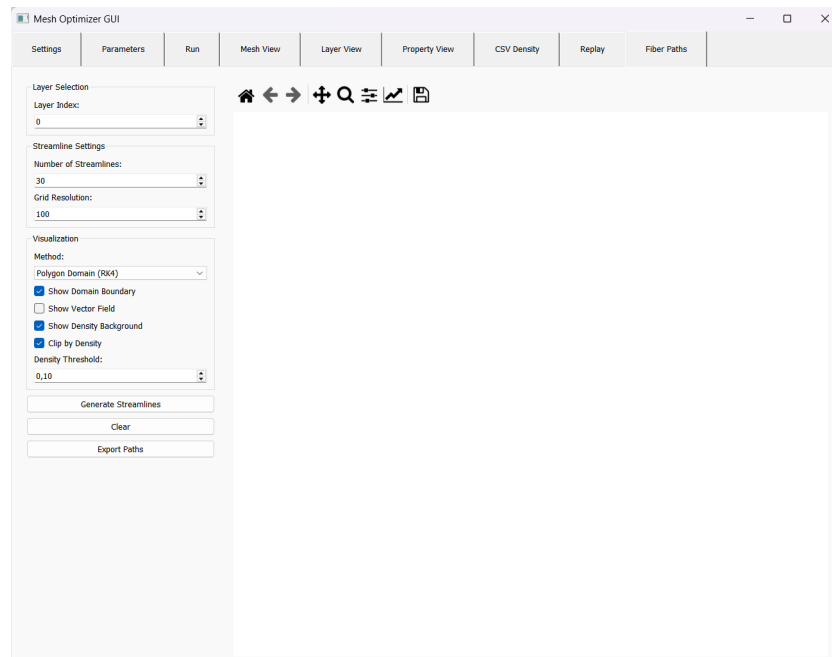


Figure 4.11: Fiber Path UI Page

The UI consists of the setup tab and the plot tab. The setup tab has a few properties to select to load the streamline: The layer number to plot, The number of streamlines and the grid resolution at which the vector field is created and the visualization of the plot where different checkboxes can be selected to show how or what is shown. To fully apply the methodology steps for the post processing a few streamline calculation functions are made which analyze and generate the streamline data.

```
generate_fiber_streamlines_polygon()
```

This function applies all the necessary calculation and runs the necessary function to retrieve the streamlines. It applies the algorithm to get to the streamlines which is shown in the methodology section. It starts by running the function which retrieves the angles from the design variables, then it builds a vector field with the near/centered method inside a domain and masks the field for a certain chosen density threshold. It then applies a seed generation function where the seeds for the streamlines are placed inside the grid and finally runs the function which generates the streamlines from the seeds using an euler stepwise method.

With the numerical implementation in place the model is now analyzed for its improvements and faults. With many different types of tests, parametric analysis, and verifications.

Table 4.32: Generate streamline function from domain and fiber orientations

Inputs	Description
element_centroids	Element centroid coordinates. Shape: (Nelem, 2)
elementy	Fiber orientation design variable
ncand	Number of candidate angles
polygon	Domain boundary polygon. If None, extracted from mesh
element_nodes	Element connectivity, needed if polygon is None
node_coords	Node coordinates, needed if polygon is None
layer_index	Which layer to visualize
elementx	Density field for clipping
density_threshold	Density threshold for clipping
resolution	Grid resolution for interpolation
n_streamlines	Number of streamlines to generate
Outputs	Description
streamlines	List of streamline arrays
X, Y	Meshgrid coordinates
vx, vy	Direction field components
domain_mask	Boolean mask for valid domain

Results, Verification and Discussion

5.1. Verification

To verify the optimizer, several tests are applied on the program, where comparisons are made with known results from the literature to verify the validity of the system. To best apply these verification methods, the tests are divided into different categories that provide a step by step validation of all the different sides of the program.

The optimizers are split into the topology optimizer and variable stiffness optimizers, to show the validity of each optimizer separately.

5.1.1. Topology Optimizer

To validate the topology optimization, several different papers are taken which use this optimization solely. In an article by Sigmund Ole [24], they show a basic algorithm for topology optimization in a rectangular design domain with the same SIMP material interpolation and a volume constraint. They optimize for a minimum compliance with additions of a filter on the sensitivity of the density variable. Together with the Karush-Kuhn-Tucker criteria which enforces the volume constraint the design is updated iteratively with a density gradient based approach. And some results are provided below.

The first test follows the following specifications:

Table 5.1: specification of rectangular plate problem

Specification	value
Design Domain	60 mm by 20 mm
Volume Fracture	0.5
penalty power	3.0
Filter radius	1.5mm
Young's Modulus	1.0
Point force	1 N

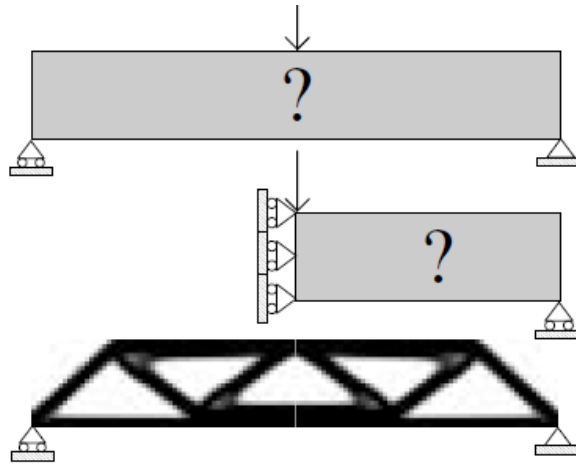


Figure 5.1: Beam problem used by Sigmund Ole [24]

To validate the program of the thesis with this test a few assumptions are made. Firstly, the material property can be approximated by setting the composite material modules all to the young's modules to create an isotropic material. Next it is assumed that a filter on the sensitivity is in some degree comparable to a filter on the design variable. The result out of the thesis is:

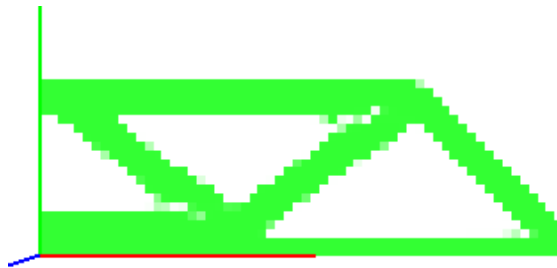


Figure 5.2: Beam problem result

The results have a high resemblance to the result from the paper although for the filter in this case the filter radius of the design variable was set to 2 as to avoid checkerboard pattern that appeared with the use of 1.5mm radius. this might be relatable to the different filter use case of sensitivity filtering vs. design variable filtering.

In the next example, the focus is on a short cantilever beam tested by S. Bulman et al.[7], where the left side is clamped and the right side has a point load down on the middle node of the part as shown in fig 5.3. The specifications of this test are:

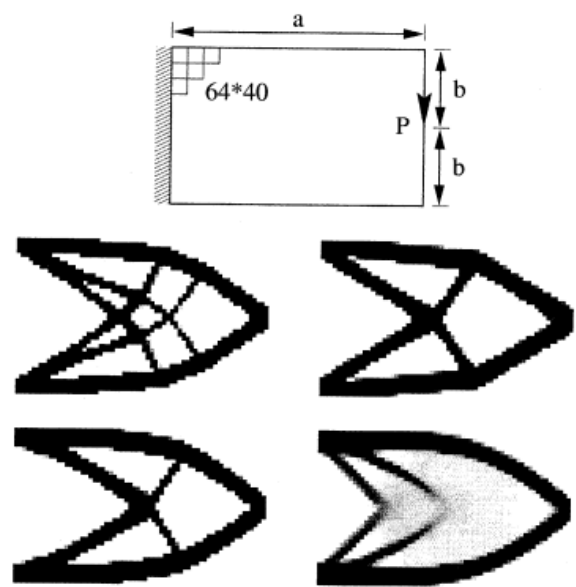


Figure 5.3: Beam problem used by S. Bulman et al.[7]

Table 5.2: specification of rectangular plate problem

Specification	value
Design Domain	16 mm by 10 mm
Element Size	0.25mm
Volume Fracture	0.375
Young's Modulus	21MPa
Point force	100 N

As a different method was used in this article the test is more as an example to see whether the program can approach the method used in the article’s results with some parameter adjustments.

The result from the program is shown in fig 5.4. It shows many similarities in the design to the results from the paper with some deviation originating from the use of a different method.



Figure 5.4: Beam problem results from 2nd test

After successfully verifying the topology optimization results trough 2 validated papers it is assumable

that the topology optimizer in the program works how it is intended and can be used later to do other tests and make other assumptions on the fact that the topology optimizer works.

5.1.2. Variable Stiffness Optimizer

Similar to the verifications done on the topology optimizer, a similar method to validate the VSO system is applied. Two articles are used with different methods and test results to verify the validity of the optimizer in the program.

Yi Lui et al.[17] have applied a similar method to the one used in the program to orientate the fiber of each element such that it results in a manufacturable fiber path result. The test retrieved from the article for validation is the first test, which represents a single layer composite laminate MBB beam. The problem to be solved is shown below in fig 5.5.

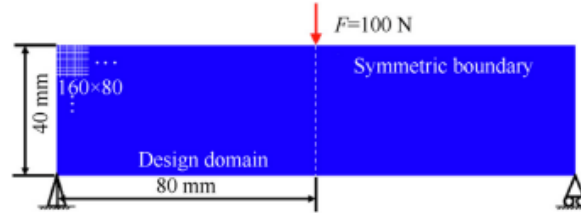


Figure 5.5: 1st test of beam problem for VSO verification

The specifications used in this problem are:

Table 5.3: specification of MBB beam problem

Specification	value
Design Domain	80 mm by 40 mm
Thickness	2mm
Element Size	1mm
Longitudinal Modulus (E_1)	39GPa
Transverse Modulus (E_2)	8.4GPa
In-Plane Shear Modulus (G_{12})	4.2GPa
Transverse Shear Modulus (G_{23})	5.0GPa
Major Poisson ratio (ν_{21})	0.26
Number of Candidates	8
Filter radius	2
Kappa Max	0.5

The resulting fiber orientations inside the design domain show good trends that follow the stress lines inside the object, and compared to the result from the program, there are some differences, however the general form out of the program follows a pattern similar enough to allow for validation. The differences can have many causes as the use of a different method for deviation adaptation through iterations, fiber filter strategy Gauss like deviations and more. The streamline plots is also shown to have follow the curvature constraint set by the program and follows similar patterns as the streamlines from the paper.

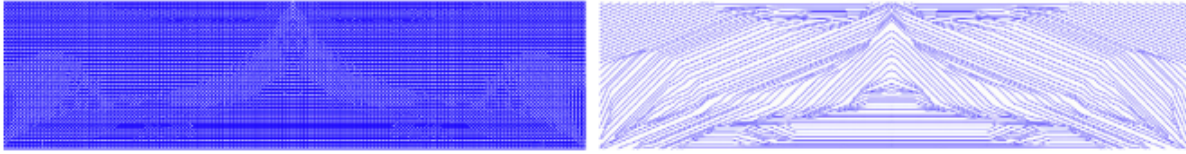


Figure 5.6: Result from paper by Yi Lui et al.[17]

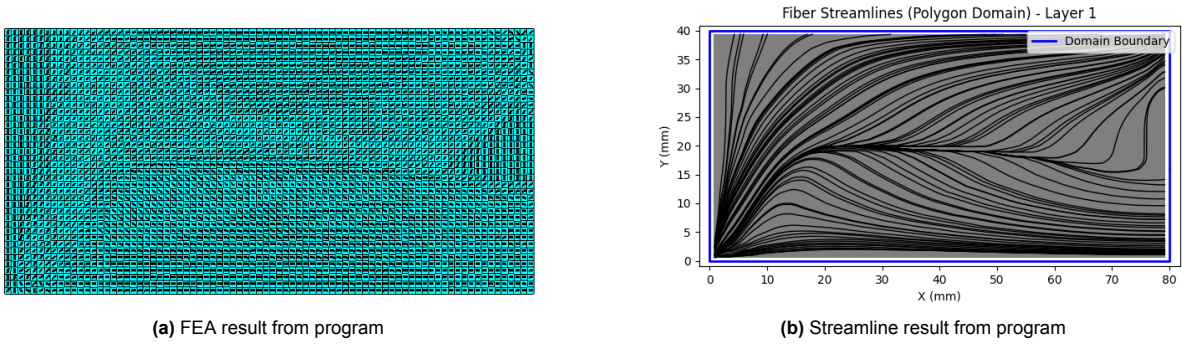


Figure 5.7: Results for 1st test of VSO verification

In another article written by C.Y. Kiyono et al.[15], similar tests on the fiber orientation optimizations are run for different models to compare. The test being run is shown in fig. 5.8.

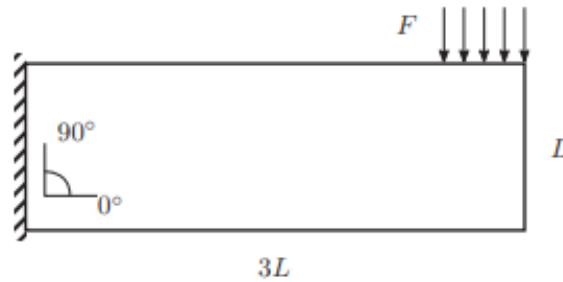


Figure 5.8: 2nd test for VSO verification used by C.Y. Kiyono et al.[15]

Here a clamped beam from the left side is under a distributed load from the right top side. The specifications of the test are shown in the table below:

Table 5.4: specification of MBB beam problem

Specification	value
Design Domain	300 mm by 100 mm
Element Size	5mm
Longitudinal Modulus (E_1)	135.0GPa
Transverse Modulus (E_2)	10.0GPa
In-Plane Shear Modulus (G_{12})	5.0GPa
Transverse Shear Modulus (G_{23})	3.0GPa
Major Poisson ratio (ν_{12})	0.3

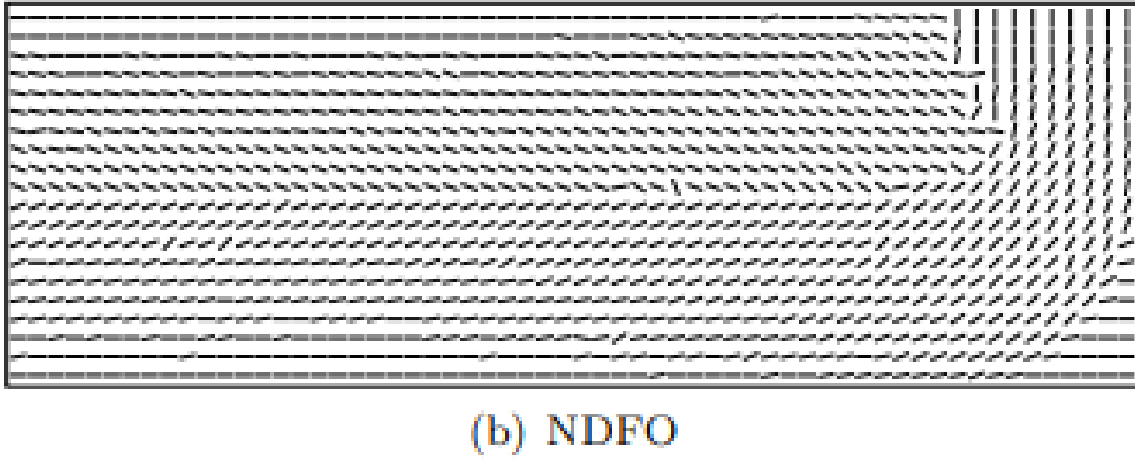


Figure 5.9: Result from paper by C.Y. Kiyono et al.[15]

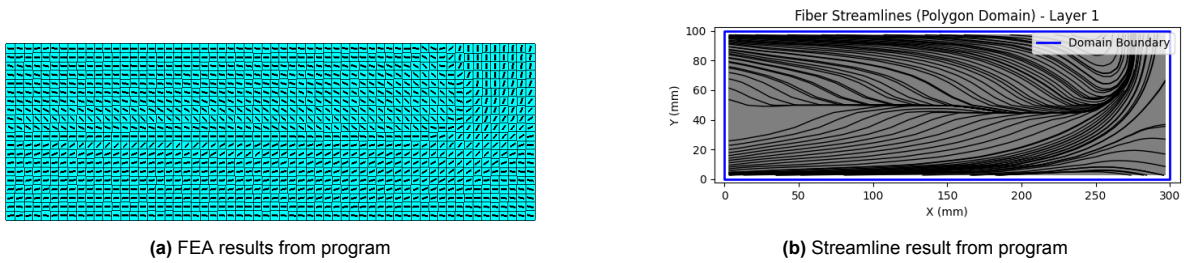


Figure 5.10: Results for 2nd test of VSO verification

The test run by the program in comparison to the result in the article show that the fiber orientations follow a similar pattern as the one provided in the article. From these two articles and test it is assumed that the fiber orientation optimizer on its own is valid enough as an optimizer for later assumptions and tests.

Next is the validation of the model as a whole. The combined approach of topology and variable stiffness optimization is checked against other literature for validation.

5.1.3. Combined Approach

Similar to the previous method, 2 literature problems are redone through the program to validate the validity of the combined approach. In This case a topology optimization using a FEA of a density SIMP method combined with different methods of DMO for the fiber orientation. In the recently mentioned article by Zunyi Duan et al.[10], a similar system to the one used in the program is used, and thus it is a good validator for the optimizer.

The first test uses a similar problem as the first test of the VSO optimizer verification, where a MBB beam is under a central load. The Setup and problem specifications can be found below:

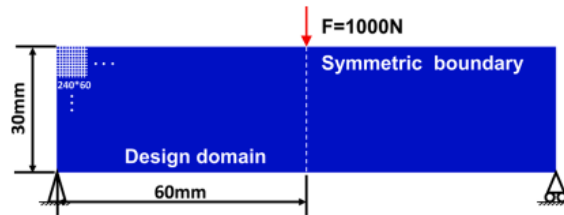
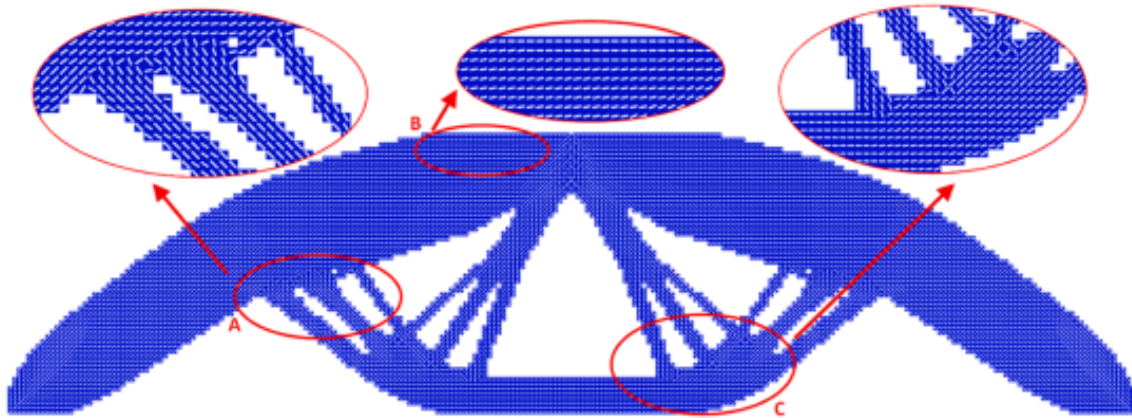
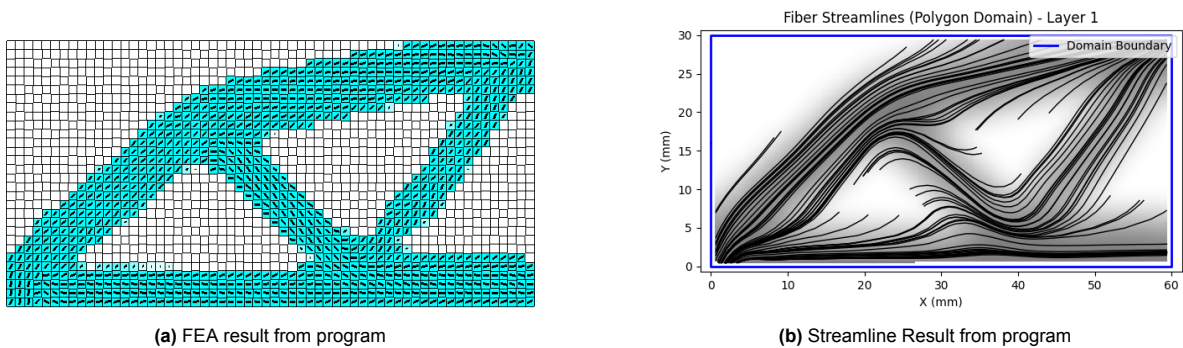


Figure 5.11: 1st test for combined optimization verification

Table 5.5: specification of MBB beam problem

Specification	value
Design Domain	60 mm by 30 mm
Thickness	2mm
Element Size	1mm
Longitudinal Modulus (E_1)	39GPa
Transverse Modulus (E_2)	8.4GPa
In-Plane Shear Modulus (G_{12})	4.2GPa
Transverse Shear Modulus (G_{23})	5.0GPa
Major Poisson ratio (ν_{12})	0.26
Number of Candidates	16
Volume Fracture	0.5

The result from the paper are very aligned with the results from the tests in that the topology is very similar in both cases however the topology in the program has thicker beam inside the component due to the higher filtering radius. The fiber-orientations similar to the result from the paper follow the trusses inside the design space aligned and provide a good degree of continuation on the splitting points.

**Figure 5.12:** Result from paper by Zunyi Duan et al.[10]**Figure 5.13:** Result for 1st test off combined optimization verification

The Second test comes from an article written by Furong Zie et al.[32], where they used different

methods from the one utilized in the thesis. The problem the article tested was a simple-supported rectangle with a central load on the top. The problem is shown below in fig 5.14.

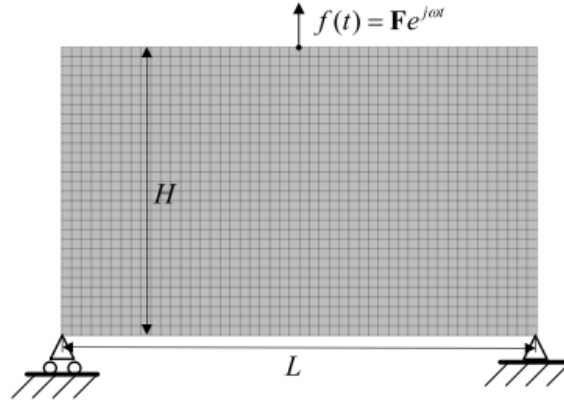


Figure 5.14: 2nd test for the combined optimization verification used by Furong Zie et al.[32]

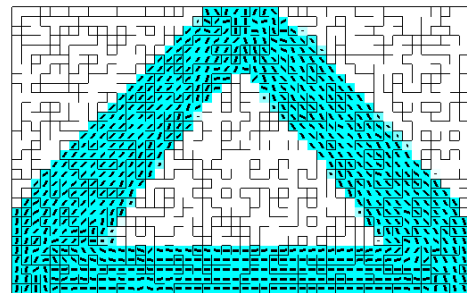
The specifications of the problem relevant for the program are also shown in the table below:

Table 5.6: specification of MBB beam problem

Specification	value
Design Domain	96 mm by 60 mm
Thickness	1mm
Element Size	1mm
Longitudinal Modulus (E_1)	130.0GPa
Transverse Modulus (E_2)	7.7GPa
In-Plane Shear Modulus (G_{12})	4.75GPa
Major Poisson ratio (ν_{12})	0.33
Volume Fracture	0.6



(a) Result from paper by Furong Zie et al.[32]



(b) Result from program

Figure 5.15: Results for 2nd test of the combined optimization verification

The comparison between the article's result and the result from the program are very similar and have a good correlation in both topology and fiber layout. Thus with both articles' problem having similar results as the program's results it is assumed that the program has a valid working combined composite laminate optimization system based on literature.

5.2. Analysis of the Parameters

After the validation of the programs, the parameters are analyzed for their effects on the result. A variation on the filter radii and power for both the topology filter and fiber orientation filters are compared respectively as well as their effect on the full optimizer model. After the element size in comparison to the design space is reviewed for better compliance results vs CPU usage.

5.2.1. Topology filter radius

To test the topology filter's radius, a singular case is taken where different radii are applied and compared to test their effectiveness and compliance result from the system.

The test utilized for this is a basic beam of dimensions, 30mm by 70mm, supported from the left with a downward force on the central right of 100N, as shown in fig 5.16. The test is run on the different radii of value, [0.5, 1, 2, 3, 5, 10, 20]. The other specifications of the test are shown in the table below:

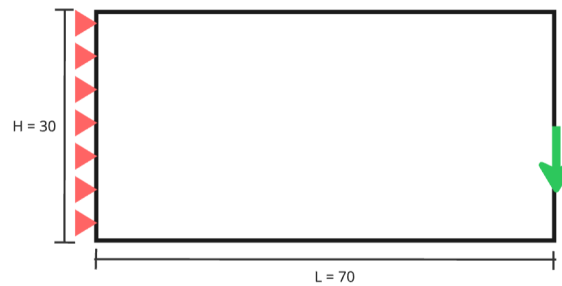


Figure 5.16: Problem setup for parameter analysis

Table 5.7: specification of rectangular plate problem

Specification	value
Design Domain	70 mm by 30 mm
Element Size	0.5mm
Volume Fracture	0.5
penalty power	3.0
Thickness	2mm
Longitudinal Modulus (E_1)	134.0GPa
Transverse Modulus (E_2)	7.71GPa
In-Plane Shear Modulus (G_{12})	4.31GPa
Major Poisson ratio (ν_{12})	0.301
Fiber Orientation	0°
P^σ	4
Compliance Convergence	1e-6

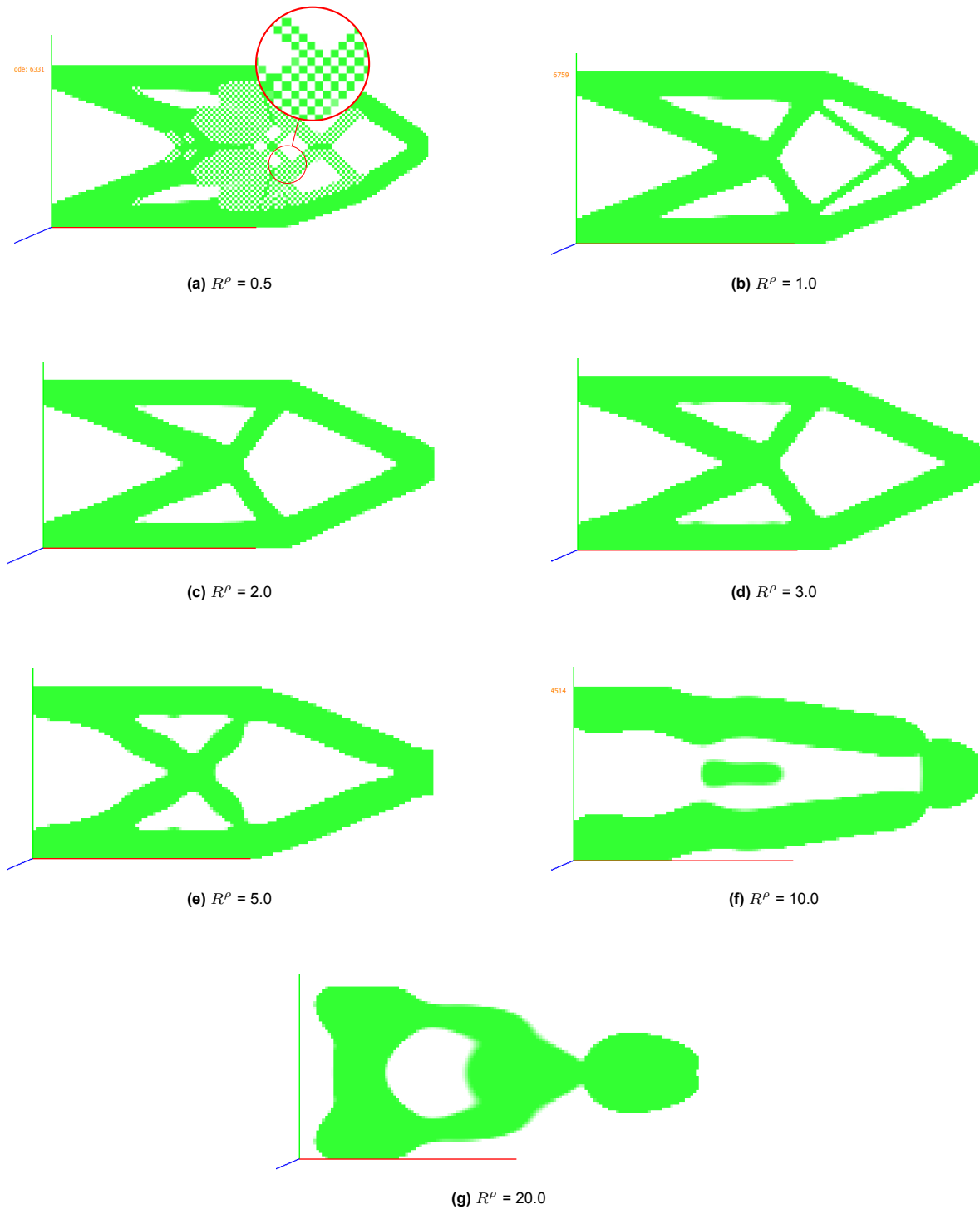


Figure 5.17: The parametrization of the density filter radius

The results from the figures show that with low radius close to the element size the results start to create areas with checker boarding patterns although the form of the final result is still somewhat similar to the desired form. For a radius of 1mm the figure shows no checker boarding patterns and 2 main truss crosses inside the object to support the load on the beam. A higher filter radius 2mm, 3mm, and 5mm removes this secondary cross structure because of the filter radius deeming the secondary cross structure less efficient when denser trusses inside the object are required. From a radius of 5mm to higher, grouping starts to form where the radius requirement for the material start to advantage circular blobs of material to structural streamlines. The effect was tested on a anisotropic material with constant

fiber orientation of 0° , to only show the effect of the filter radius on the object. When trying with different constant orientation field, different forms of the final result might be seen but the effect of the filter radius remains.

The following table also shows the resulting compliance of each of the tests where a compliance convergence of $1e-6mJ$ was used or the maximal iteration of 200 as to not do unnecessary overruns for this test. The convergence at the 200th iteration in all cases where this is applied is also low enough to be admirable.

Table 5.8: Compliance change on filter radius

Radius	Compliance	Compliance Convergence	Iteration	CPU time
0.5	7.491519	$1e-6$	94	196s
1.0	7.768151	$1.6e-5$	200	388s
2.0	8.269382	$1e-6$	161	256s
3.0	9.021899	$6.9e-6$	200	331s
5.0	10.940971	$9.7e-6$	200	337s
10.0	16.234241	$1e-6$	120	190s
20.0	25.455565	$1e-6$	131	216s

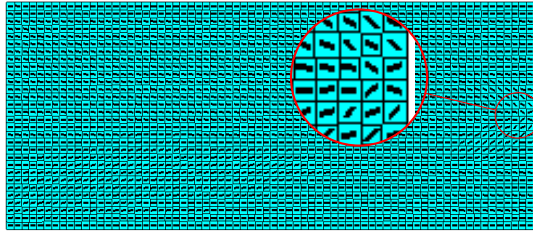
The compliance as expected go up with higher filter radii due to the decrease in problem complexity. where in a theoretical sense checkerboard patterns for a simp method offer the strongest systems.

5.2.2. Fiber Orientation filter radius

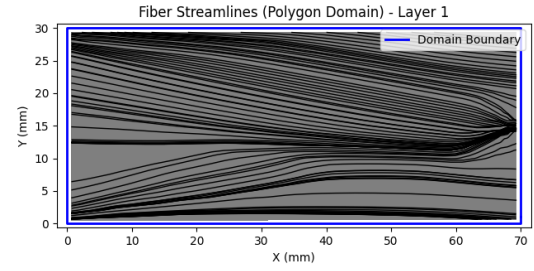
For the filtering test of the fiber orientation design variable, a similar test is taken as used in the topology filter test. The same problem is analyzed and the specifications of this problem follow:

Table 5.9: specification of rectangular plate problem

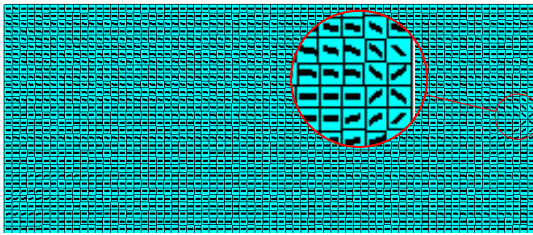
Specification	value
Design Domain	70 mm by 30 mm
Element Size	1mm
penalty power	3.0
Thickness	2mm
Longitudinal Modulus (E_1)	134.0GPa
Transverse Modulus (E_2)	7.71GPa
In-Plane Shear Modulus (G_{12})	4.31GPa
Major Poisson ratio (ν_{12})	0.301
P^{σ}_{init}	4
α	0.1
filter power	2
κ_{max}	0.5 mm^{-1}
Compliance Convergence	$1e-6$



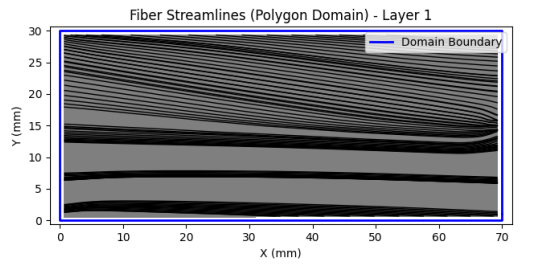
(a) FEA result



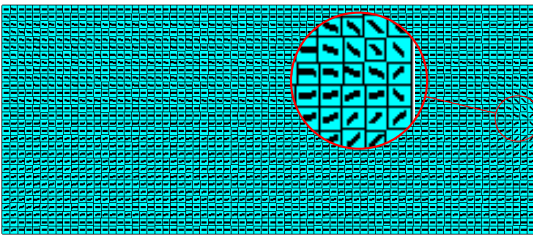
(b) Streamline result with zoom on checkerboard pattern

Figure 5.18: Fiber orientation filter radius $R^\theta = 1.0$ 

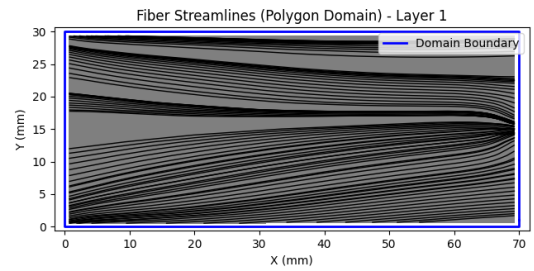
(a) FEA result



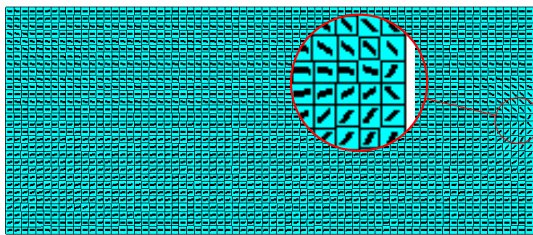
(b) Streamline result

Figure 5.19: Fiber orientation filter radius $R^\theta = 2.0$ 

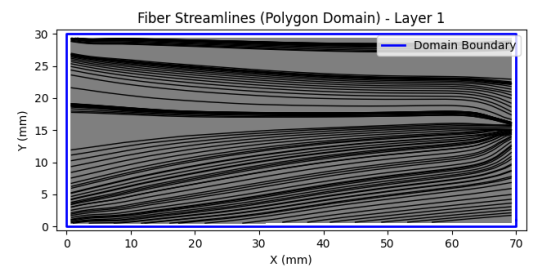
(a) FEA result



(b) Streamline result

Figure 5.20: Fiber orientation filter radius $R^\theta = 4.0$ 

(a) FEA result



(b) Streamline result

Figure 5.21: Fiber orientation filter radius $R^\theta = 5.0$

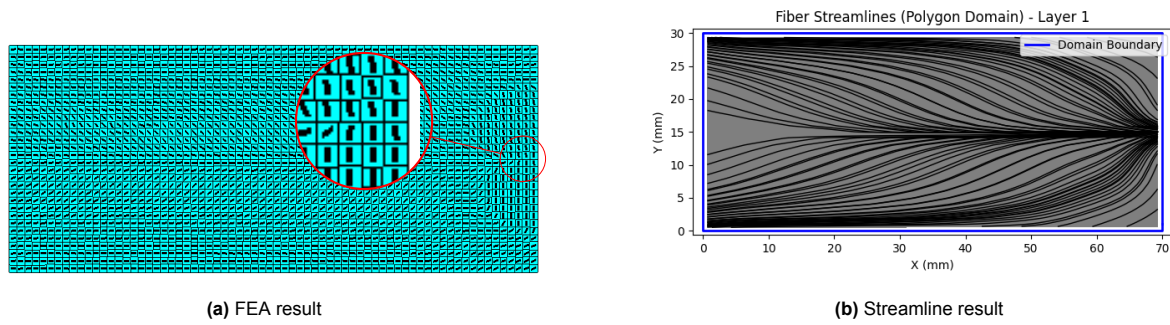


Figure 5.22: Fiber orientation filter radius $R^\theta = 10.0$

The results show that for the lower radii, there is high amount of checker boarding. Checker boarding refers to a crossing pattern over a 4 element pair as shown below in fig 5.23. For the results of radii 2 to 5mm there is increased smoothing around high stress areas however there is still some degree of loss in uniform direction patterns because the neighbors are less influential on each other. This pattern disappears with the 10 radius as a more general uniform streamline directions are observed and a better conclusion towards higher stress areas, and where the checkerboard patterns start to disappear.

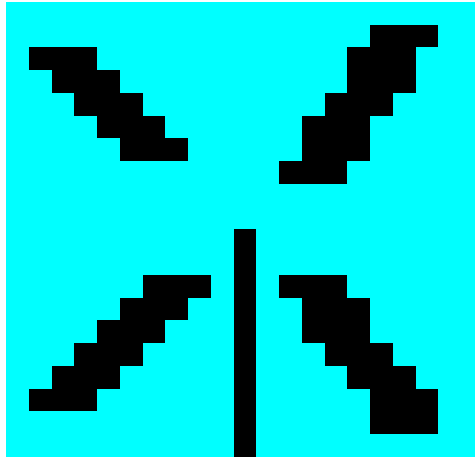


Figure 5.23: Checkerboard pattern for the fiber orientation optimizer

The compliance, found in the table below, follows a strange pattern where at first an increase in radii provides an increase in compliance due to the added safety of the problem which is similar to filling up the found minima in a sense. However the case of the filter radius of 10mm shows a drop in the compliance. This is due to the fact that the radius is so big it skips over the previously found minima dip and settles into a different minima with an improved compliance.

Table 5.10: Compliance change on filter radius

Radius	Compliance	Compliance Convergence	Iteration	CPU time
1.0	3.529934	$1e-6$	38	178s
2.0	4.208376	$1e-6$	36	218s
3.0	4.070932	$1e-6$	55	563s
5.0	4.365359	$1e-6$	48	572s
10.0	3.224601	$1e-6$	59	629s

From these result the 10x the element size filter radius showed the best results for clean gradual streamlines and will thus be used further in the analysis.

5.2.3. Fiber Orientation filter power

Now maintaining the filter radius of 10 and adapting it to different filter power levels is the next test to fine-tune the fiber orientations curvatures better,

So once again the same problem is used, the only part that changes is a through the test a continuous fiber orientation filter radius of 10mm is subjected to the power level changes of [1, 2, 3, 5] to evaluate it's effect on the resulting design.

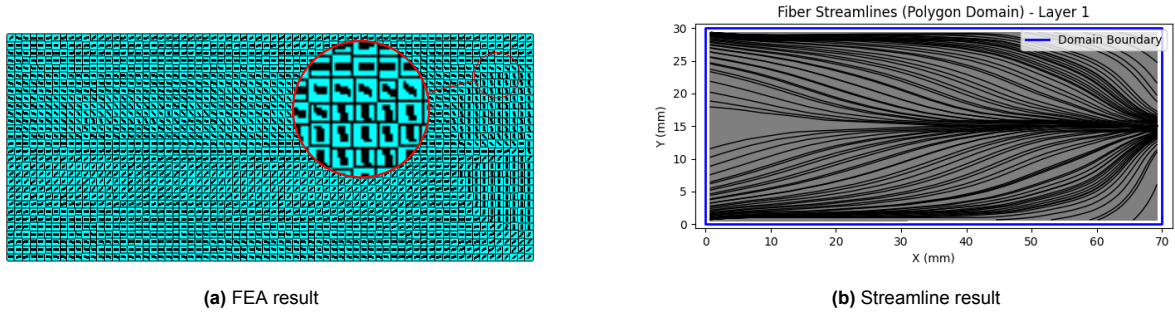


Figure 5.24: Fiber orientation filter power $p^\theta = 1.0$

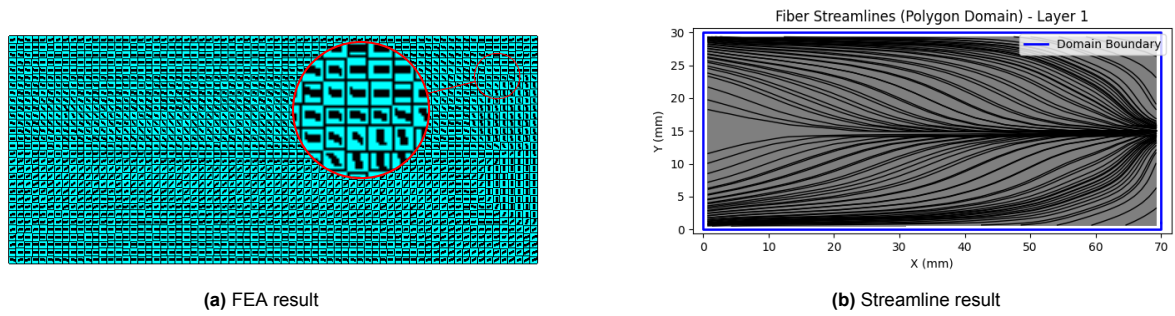


Figure 5.25: Fiber orientation filter power $p^\theta = 2.0$

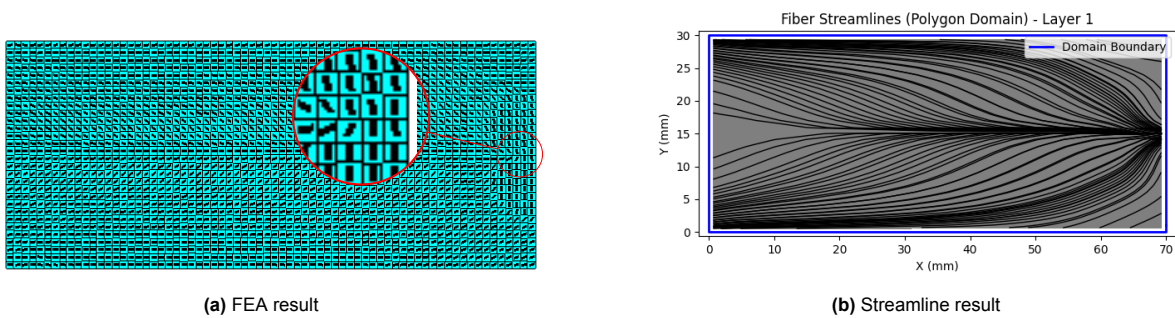


Figure 5.26: Fiber orientation filter power $p^\theta = 3.0$

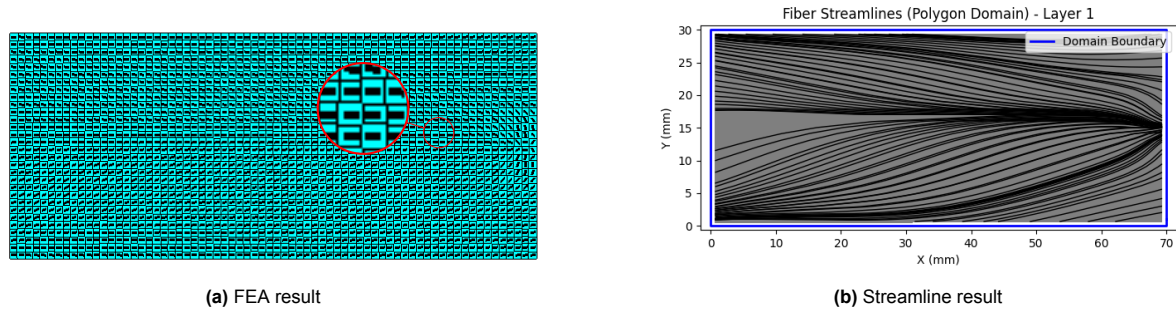


Figure 5.27: Fiber orientation filter power $p^\theta = 5.0$

The weighting of radius 5 with power of 1 show signs of high density groups of fiber orientation in fixed direction and the degree of change in that group is low. This expected as a low power makes more fiber in a certain area dependent on each other with a linear change. Increasing the power allows for more drastic change in an area and thus steeper curvature changes as shown for a power of 2 and 3. However at very high power values, power 5, the dependency on neighbors start to fail as a whole where a direct neighbor strongly follows the fiber orientation of the element unless overpowered by the heavy need for change to improve the compliance. This however causes an increase in compliance as shown below. Later with the combined parametric analysis a better use case for the power might also be shown but for now a power of 2 is assumed as it shows the best result for both keeping somewhat the curvature with the orientations but allowing some fast changes if necessary.

Table 5.11: Compliance change on filter radius

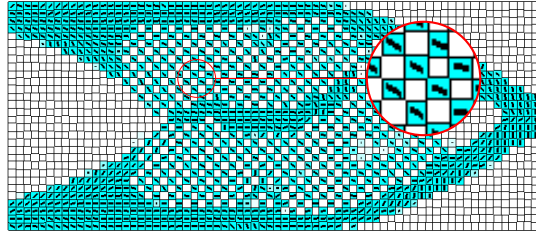
Power	Compliance	Compliance Convergence	Iteration	CPU time
1.0	3.354017	$1e - 6$	23	147s
2.0	3.224601	$1e - 6$	59	678s
3.0	3.092400	$1e - 6$	26	182s
5.0	3.423126	$1e - 6$	45	301s

5.2.4. Combined effects

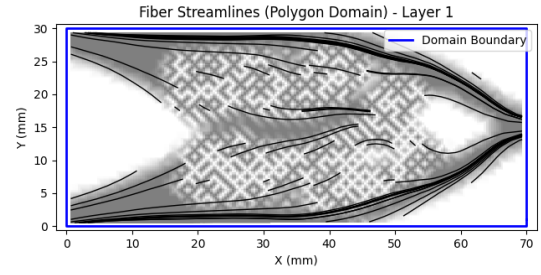
The combined results from the radii and power in the case of the full optimizer is also interesting to see as the points of interest for fiber orientations change with the topology and vice versa. To view this 3 cases are taken, Topology optimization and VSO optimization with density filtering but without fiber orientations filtering. The second case follows the same model however with fiber filtering. Finally, the third case follows the effect of the change in power with topology and fiber filtering and optimizations.

Topology Optimization and filtering with VSO

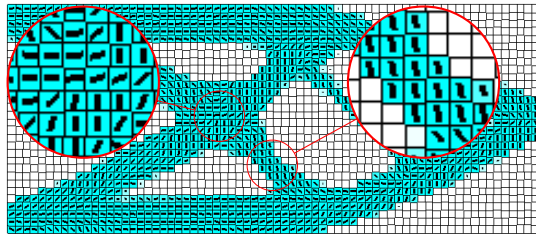
For this test 4 cases are generated again for 4 different radii. Because as shown before a radius higher than 5 will result in area blobs forming which is not wanted. So here the radii [1, 2, 3, 5] are tested with the same specifications as before and no fiber filtering.



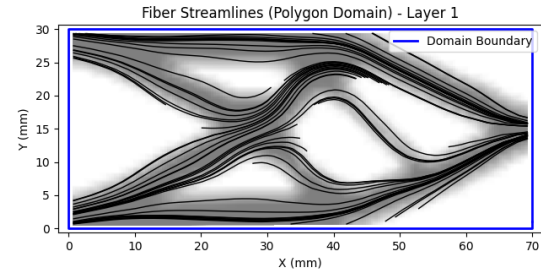
(a) FEA result



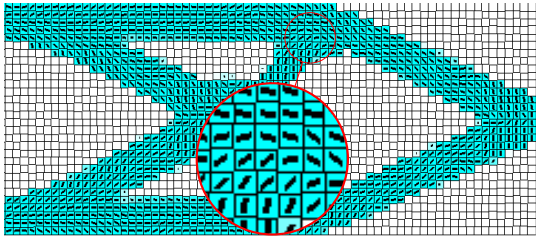
(b) Streamline result

Figure 5.28: Combined approach with density filter $R^\rho = 1.0$ 

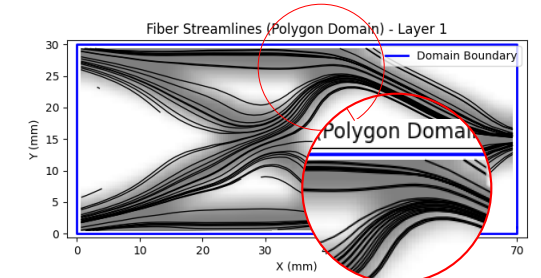
(a) FEA result



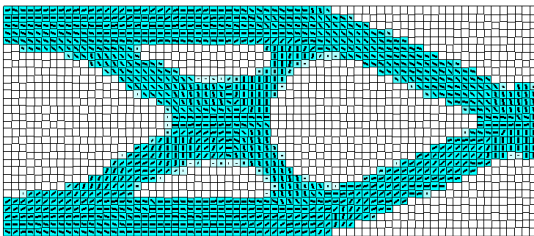
(b) Streamline result

Figure 5.29: Combined approach with density filter $R^\rho = 2.0$ 

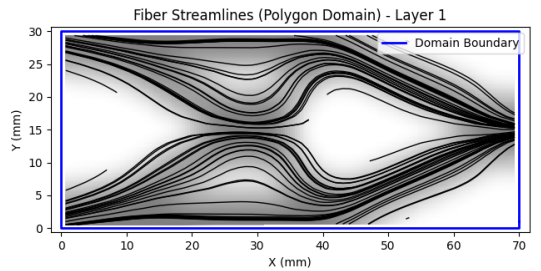
(a) FEA result



(b) Streamline result

Figure 5.30: Combined approach with density filter $R^\rho = 3.0$ 

(a) FEA result



(b) Streamline result

Figure 5.31: Combined approach with density filter $R^\rho = 5.0$

The results show that the density checkerboard pattern decreases when the radius goes up and the fiber orientations follow the directions of the trusses inside the design domain. Also with the 3x element size radius and 5x element size radius the small extra trusses disappear and the main trusses become

thicker and the fiber orientation keep following these systems, however, the crossing points are still too direct on the fiber orientations. The compliance also follows a similar pattern as observed with the previously testes density radii as shown in the table below, where a higher radius leads to a higher compliance as you are limiting the possible results. However what is also interesting in the table is that the time to convergence rises in the case where checkerboarding is allowed as the optimizer has more possibilities to play with and has a harder time stabilizing on a result. Once a more limiting density radius comes in it follows the more standard time patterns where a higher radius has more difficulty setting up the correct fiber alignments within bigger areas.

Table 5.12: Compliance change on filter radius

Radius	Compliance	Compliance Convergence	Iteration	CPU time
1.0	5.698571	$1e-6$	87	1138s
2.0	6.000746	$1e-6$	59	577s
3.0	6.187996	$1e-6$	56	603s
5.0	7.799981	$1e-6$	62	721s

In this test a filter radius following 3x the element size provides the best results in removing checker-board patterns or isolated elements and at the same time not cause too much area blobbing. Thus for the following parts of the thesis a radius of 3x the element size is adapted.

Topology optimization with filtering and VSO with fiber orientation filtering

Next an analysis is run on the fiber filter radius with a fixed density filter radius of 3mm as explained in the previous result. The power is set to 2 and the tested radii are [1, 2, 3, 5, 10] as they showed the best results in the previous test.

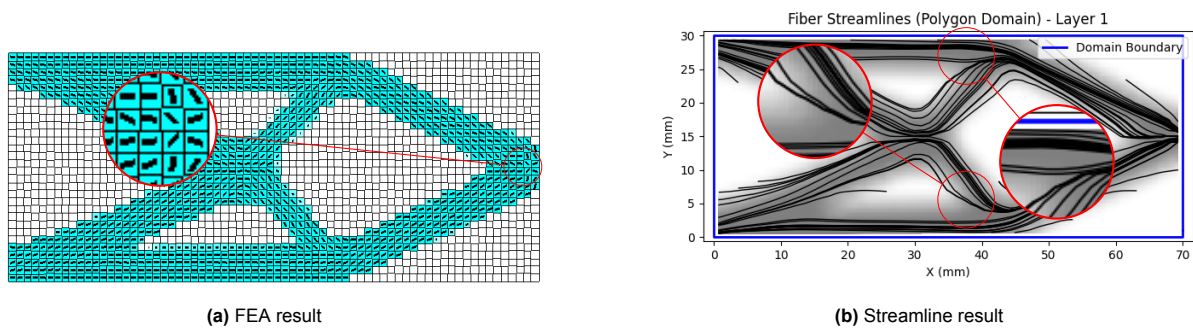


Figure 5.32: Combined approach with fiber orientation filter $R^\theta = 1.0$

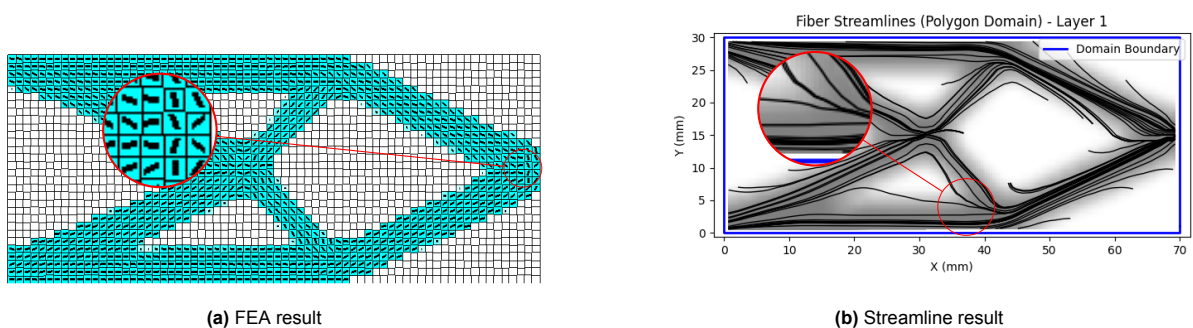


Figure 5.33: Combined approach with fiber orientation filter $R^\theta = 2.0$

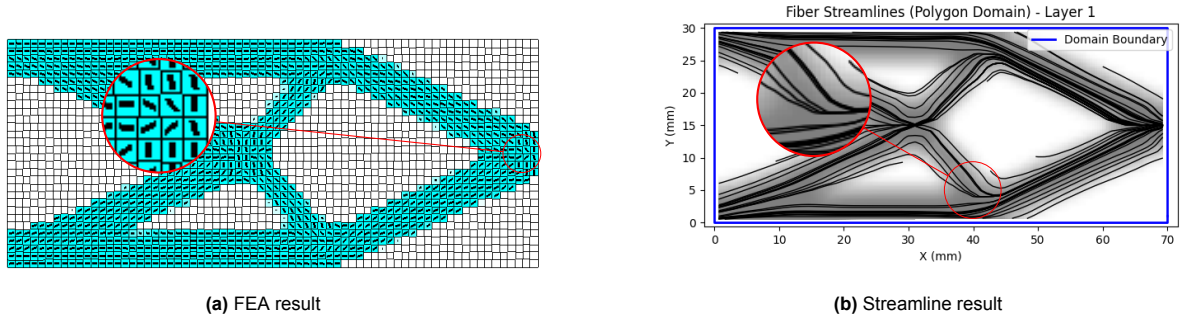


Figure 5.34: Combined approach with fiber orientation filter $R^\theta = 3.0$

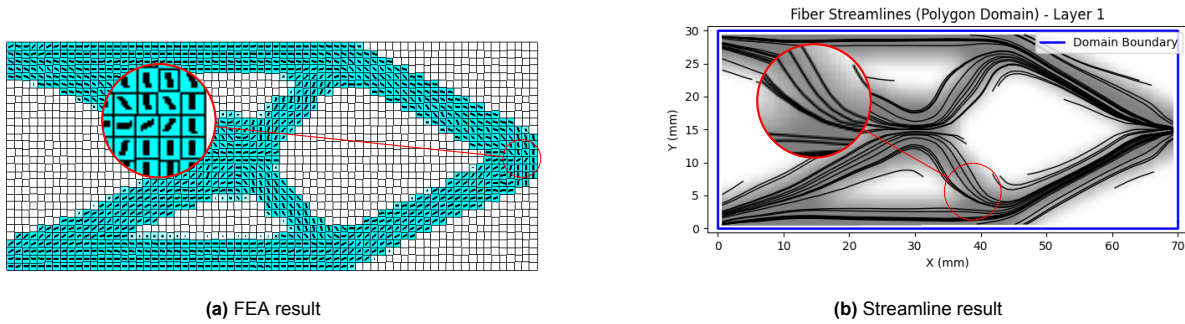


Figure 5.35: Combined approach with fiber orientation filter $R^\theta = 5.0$

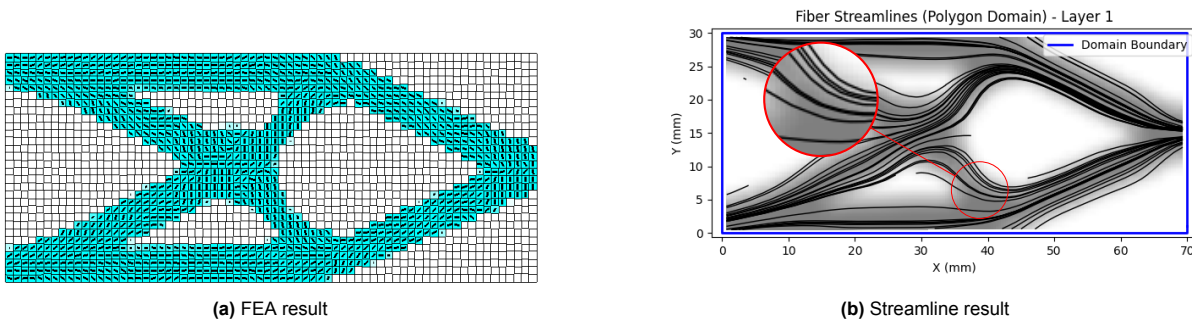


Figure 5.36: Combined approach with fiber orientation filter $R^\theta = 10.0$

As expected from the previous tests the results show that the radii 1 to 5 have signs of more disturbed streamlines which become more smooth the higher the radius. The radius of 5 has already a good degree of uniformity in the streamlines but still lacks it in certain locations. Also the checker boarding patterns are found in the results from the radii 1 to 3 but gets more rare and only visible in higher stress location the higher the radii and is almost not found in the radii 5 and 10. What is also visible is that with the radii of 10 in this case the truss is trying to cross over because of this push towards a uniform areas in 1 direction. While it does show a smoother result it shows that it limits the optimizer into certain results.

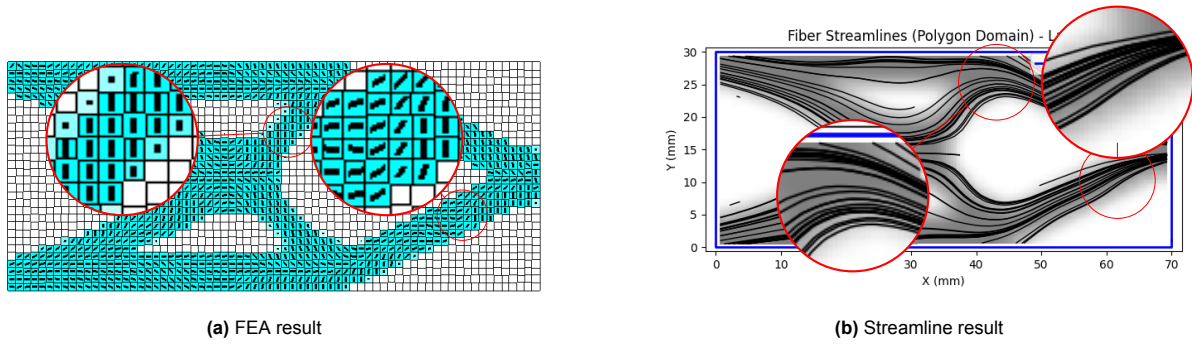
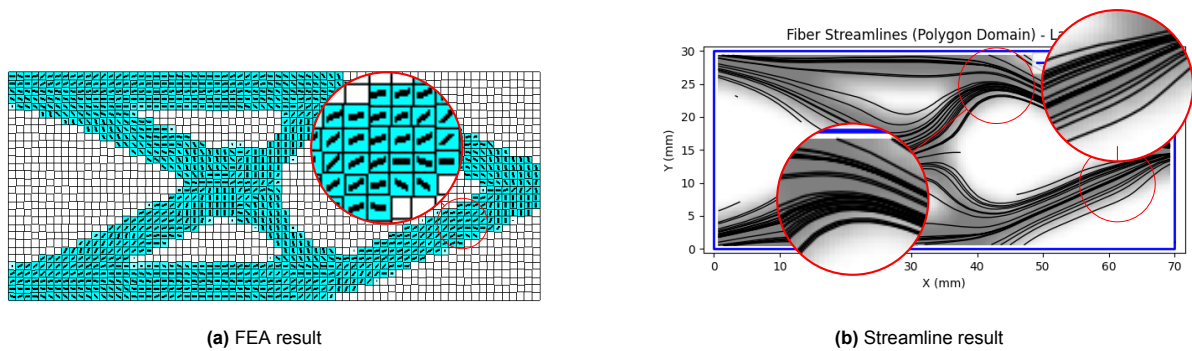
The table above shows the compliance of the results and it is clear as previously noted that a higher radius for the fiber filter decreases the optimizer's control and thus increases the resulting compliance in a general sense as seen for radii 3 to 10. The radii 1 and 2 show a more different compliance result likely because the results falls into this local minima due to the non convexity and complexity added by the curvature constraint. For future optimizations a fiber filter radius of 5 is held as it showed the most promise for manufacturability and smooth streamlines even at the cost of the increased compliance.

Table 5.13: Compliance change on filter radius

Radius	Compliance	Compliance Convergence	Iteration	CPU time
1.0	5.459752	$1e-6$	78	741s
2.0	5.229438	$1e-6$	56	375s
3.0	4.966077	$1e-6$	62	513s
5.0	5.936250	$1e-6$	53	512s
10.0	6.239977	$1e-6$	83	838s

Power test for VSO and Topology optimization with all active filters

The previous power test is also done again for the combined approach to examine the effectiveness of the power on certain stress crossing points and the smooth curvature for those locations. The same problem was set with a density filter radius of 3mm a fiber filter radius of 10mm as it provides a higher range to test the power effect on. The different powers are tested in the range of $[1, 2, 3, 4, 5]$ as any higher power did not show high differences in the previous tests.

**Figure 5.37:** Combined approach with fiber orientation filter power $p^\theta = 1.0$ **Figure 5.38:** Combined approach with fiber orientation filter power $p^\theta = 2.0$

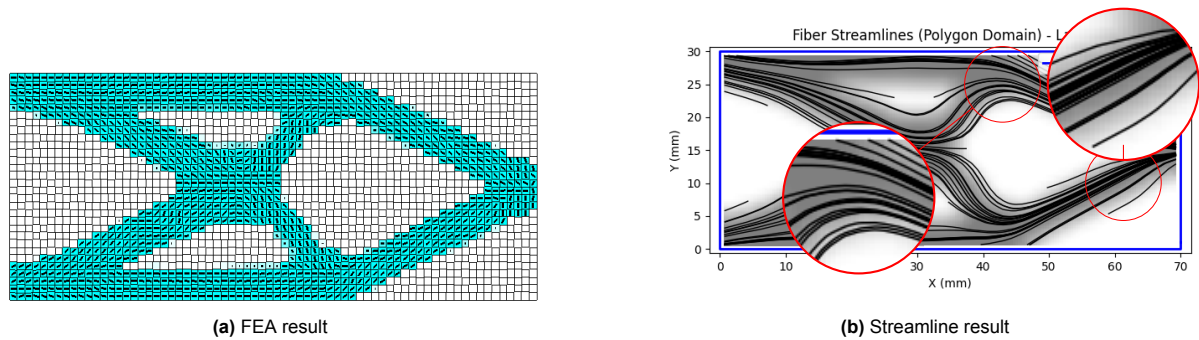


Figure 5.39: Combined approach with fiber orientation filter power $p^\theta = 3.0$

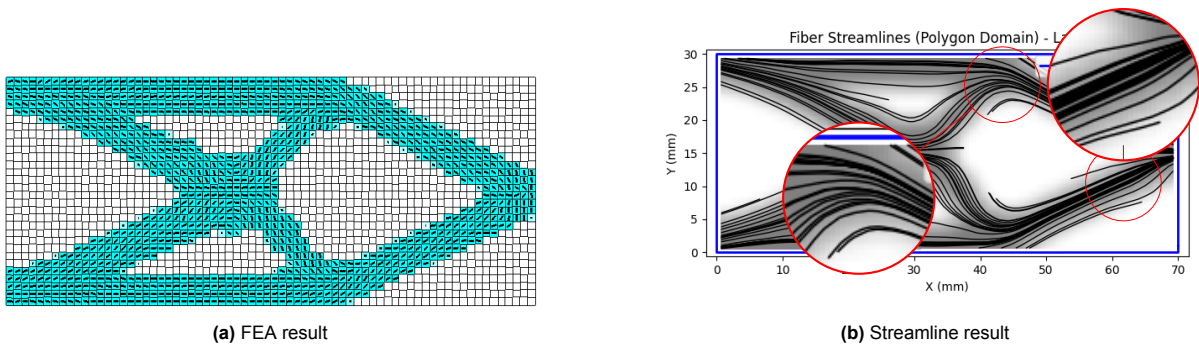


Figure 5.40: Combined approach with fiber orientation filter power $p^\theta = 4.0$

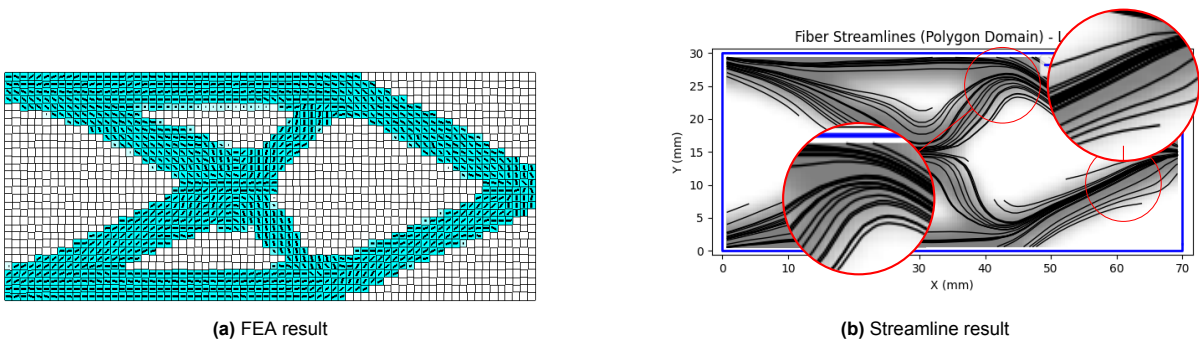


Figure 5.41: Combined approach with fiber orientation filter power $p^\theta = 5.0$

Table 5.14: Compliance change on filter radius

Radius	Compliance	Compliance Convergence	Iteration	CPU time
1.0	7.374553	$1e-6$	76	1018s
2.0	6.239869	$1e-6$	70	639s
3.0	6.310682	$1e-6$	73	609s
4.0	6.203482	$1e-6$	67	652s
5.0	6.093556	$1e-6$	80	836s

The results show that for a power of 1 the fiber orientations is pushed too much towards this uniform streamline where the fibers should be straight and show more wavy patterns to reach the end. The

power of 2 case shows already a better increase in straight streamlined trusses on top of better control and transition on junction points of the trusses. For power levels of 3 and higher there are still good transitions at the junction, all though they happen more abruptly and does not smooth into the other truss. To keep some well streamlined fiber orientation in the resulting designs while keeping them manufacturable a power level of 2 is used in the rest of the thesis.

5.3. Random Initialization points

Because the problem is heavily non-convex a comparison is done on a mix of randomized starting design space systems to test if the retrieved result from the set initialized position is better or worse than that of some randomized position results. To test this the previous problem is used again with the parameters chosen as set below with a fiber radius of 10mm to push the fiber orientations towards a uniform design. Also a few tests are done with a different curvature constraint scaling parameter to see the effect of it on those randomized initial point because a higher scaling parameter should allow for a more gradual result as the curvature constraint becomes more strongly set. Same as with the penalty power for the density parameter a gradual increase to 3 from 1 is tested to gradually bring the design to the desired result. To set up the random design variable initial values the code is adjusted in the object file, in the function `initialize_element_x_y`, where the `self.Elementx = initial_x * np.ones((self.Nelem, self.NLayers))` line is switched by a randomizer function with the limits of the design variables `self.Elementx = np.random.uniform(0.01, 1, size=(self.Nelem, self.NLayers))`, this switch is also applied for the fiber orientation design variable.

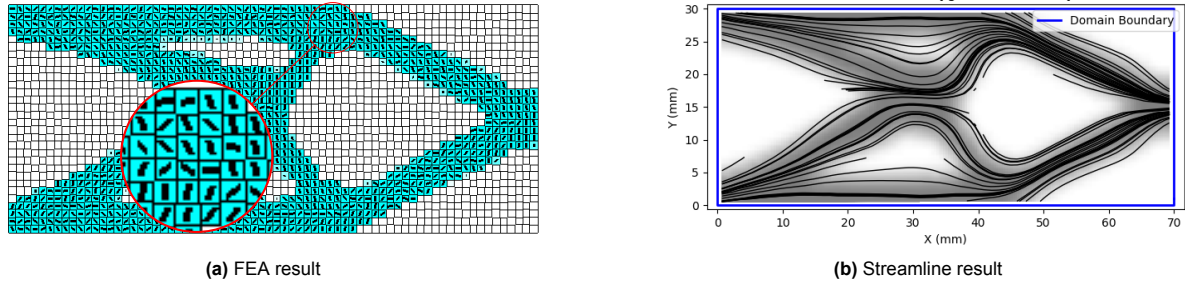
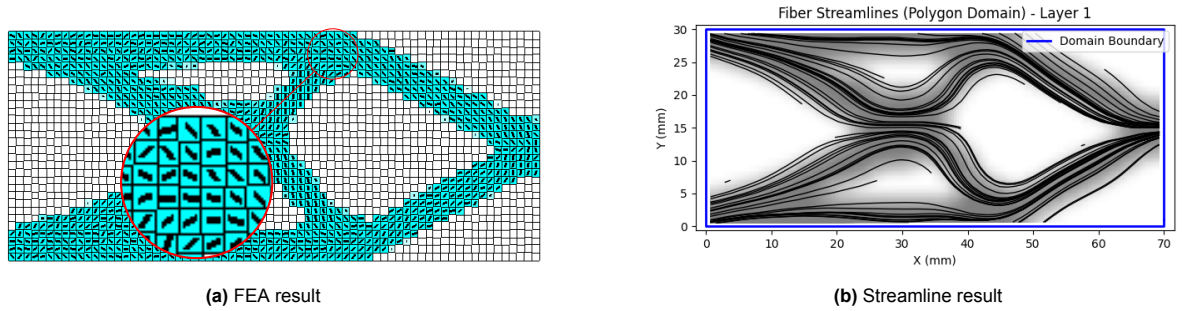
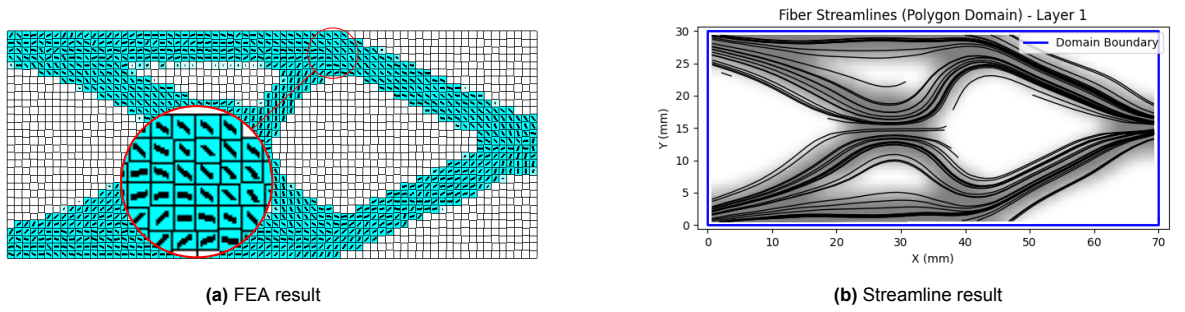
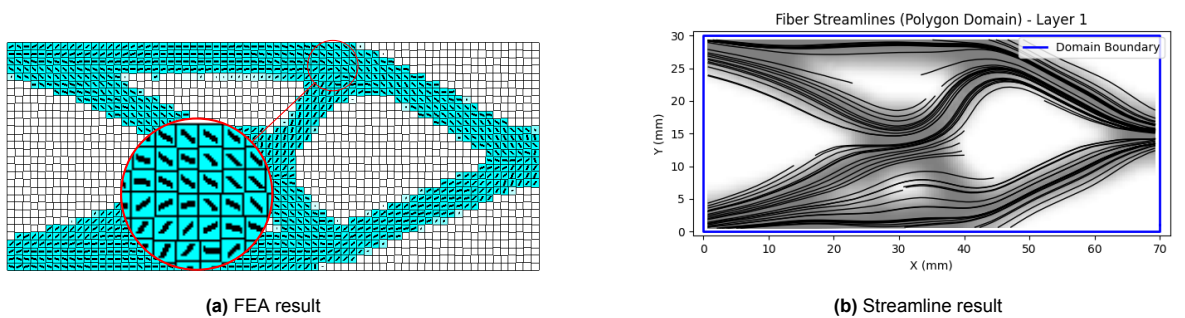
Specifications

Table 5.15: specification of rectangular plate problem

Specification	value
Design Domain	70 mm by 30 mm
Element Size	1mm
Volume Fracture	0.5
Thickness	2mm
Longitudinal Modulus (E_1)	134.0GPa
Transverse Modulus (E_2)	7.71GPa
In-Plane Shear Modulus (G_{12})	4.31GPa
Major Poisson ratio (ν_{12})	0.301
Compliance Convergence	1e-6
Density Filter Radius	3mm
Fiber orientation filter radius	10mm
Fiber orientation filter power	2
κ^{max}	$0.5mm^{-1}$
k_{max}	50

Results

The test is run for each curvature constraint scaling parameter of $[0.00001, 0.0001, 0.001, 0.01]$ to see the effect of it. The maximum amount of iterations where the initial penalty deviation, k_{max} , stays constant is also increased to 50 to leave some time for the topology and fiber orientation to settle before pushing the element properties towards the one fiber orientation. The results are shown in Figure 5.42 to 5.45:

Figure 5.42: Random initialization test for $\mu_1 = 0.00001$ Figure 5.43: Random initialization test for $\mu_1 = 0.0001$ Figure 5.44: Random initialization test for $\mu_1 = 0.001$ Figure 5.45: Random initialization test for $\mu_1 = 0.01$

In general, these results show that the randomization effect in general has a greater effect on the fiber orientation than the density. This is because the fiber orientation effect the stiffness tensor orientation directly. Their problem is more non-convex than the topology problem which has a consequence of

Table 5.16: Compliance change on filter radius

scale	Compliance	Compliance Convergence	Iteration	CPU time
0.00001	6.626760	$1e-6$	81	2159s
0.0001	7.141726	$1e-6$	94	2980s
0.001	6.965068	$1e-6$	86	1321s
0.01	6.715379	$1e-6$	94	1736s

easier locking into a local minimum which results in these high variations of the fiber orientation. This is solved to some degree as is shown with an increased scaling of the curvature constraint as it pushes the fibers to follow the curvature to a stronger degree and not allow for any abrupt curves.

5.4. CBeam

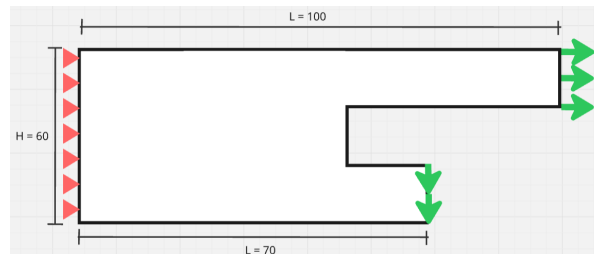
The Next section show the example of a CBeam with dual loading conditions. The CBeam uses material properties as shown below:

Table 5.17: CBeam Material Properties

Specification	value
Longitudinal Modulus (E_1)	134.0GPa
Transverse Modulus (E_2)	7.71GPa
In-Plane Shear Modulus (G_{12})	4.31GPa
Major Poisson ratio (ν_{12})	0.301

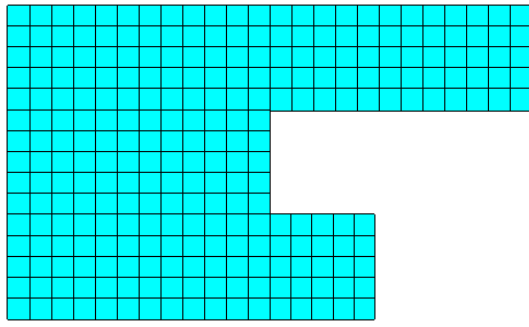
Problem definition

The C Beam problem is defined using the figure below:

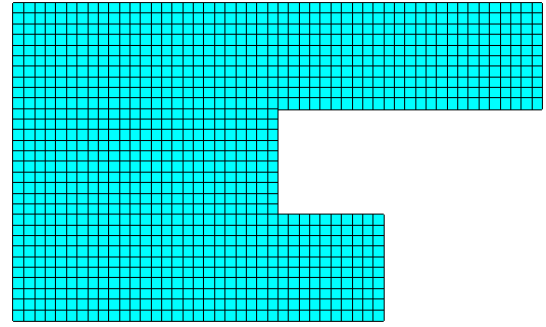
**Figure 5.46:** CBeam Problem Definition

Here the left side of the CBeam is clamped with 0 displacement boundary conditions. The upper part of the beam has a traction force outwards of 50N/mm and the lower part of the beam has a traction force downwards of 50N/mm.

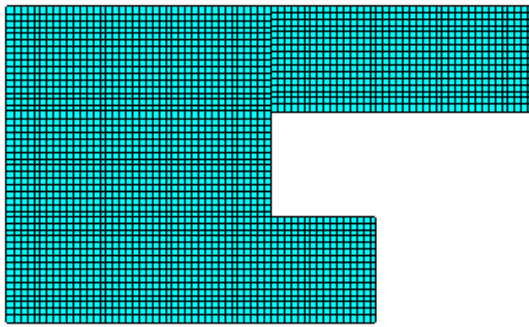
Mesh generation The problem is run for 4 different element-sizes, 1mm/1,25mm/2mm/4mm as shown below:



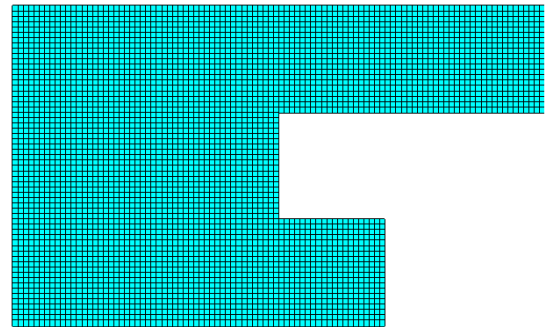
(a) Mesh generation with elementsize 4mm



(b) Mesh generation with elementsize 2mm



(c) Mesh generation with elementsize 1,25mm



(d) Mesh generation with elementsize 0.5mm

Figure 5.47: CBeam Mesh Generations

The problem is then run with the previously retrieved simulation parameters from section 5.2:

Table 5.18: specification of rectangular plate problem

Specification	value
Volume Fracture	0.5
Maximum curvature	0.5 mm^{-1}
Thickness	2mm
Compliance Convergence	1e-6
Density Filter Radius	3x Elementsize
Fiber orientation filter radius	5x Elementsize
Fiber orientation filter power	2
k_{max}	30
P_{init}^{σ}	4.0
α	0.1
γ_{init}	1
γ_{max}	3
$\tilde{\gamma}$	0.1

It is expected for the finer mesh to provide a more detailed and precise result on top of a lower compliance as the possibilities in the optimization are higher. With this is expected to come a larger gain in CPU time as more and more elements can drastically increase the time it takes to optimize for each generation. The results from the tests are shown below:

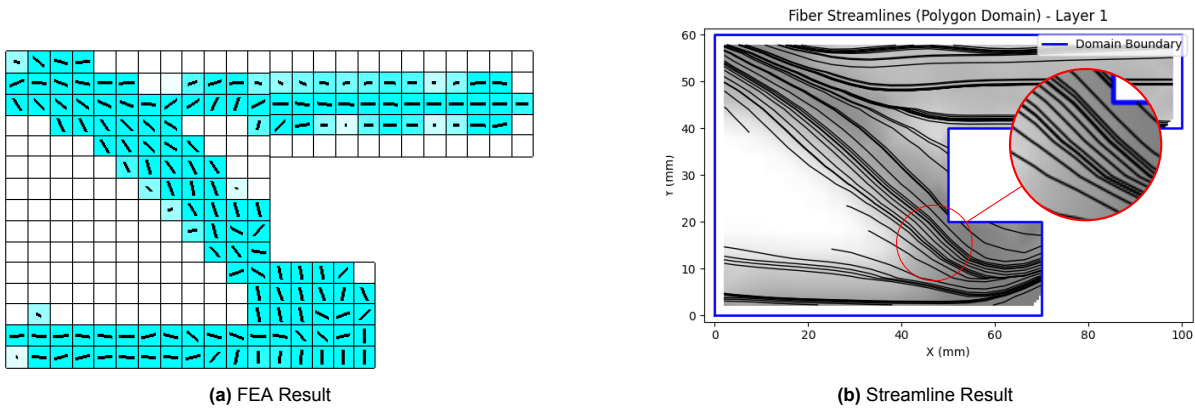


Figure 5.48: Mesh generation with elementsize 4mm

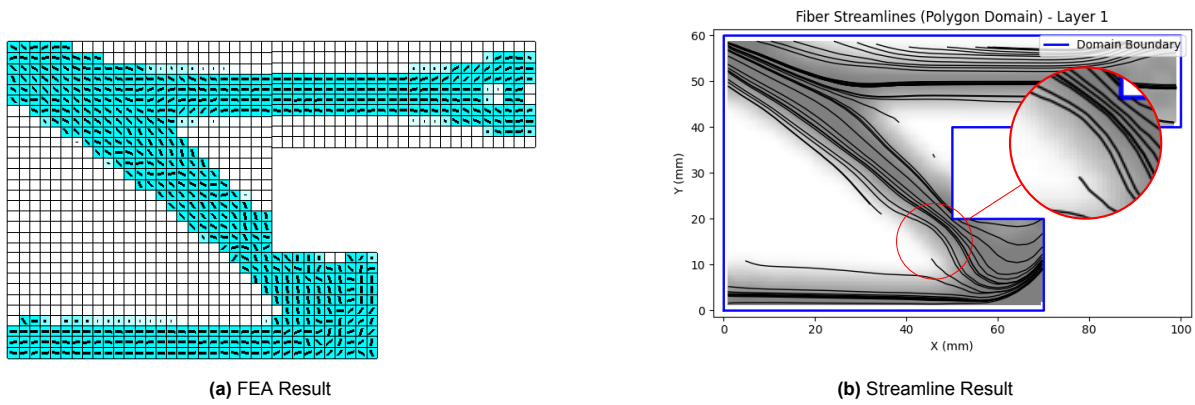


Figure 5.49: Mesh generation with elementsize 2mm

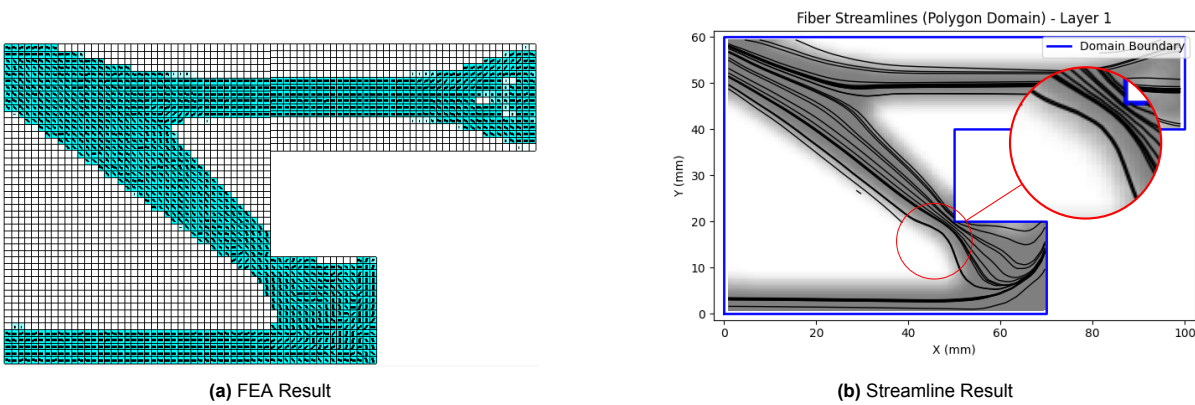


Figure 5.50: Mesh generation with elementsize 1,25mm

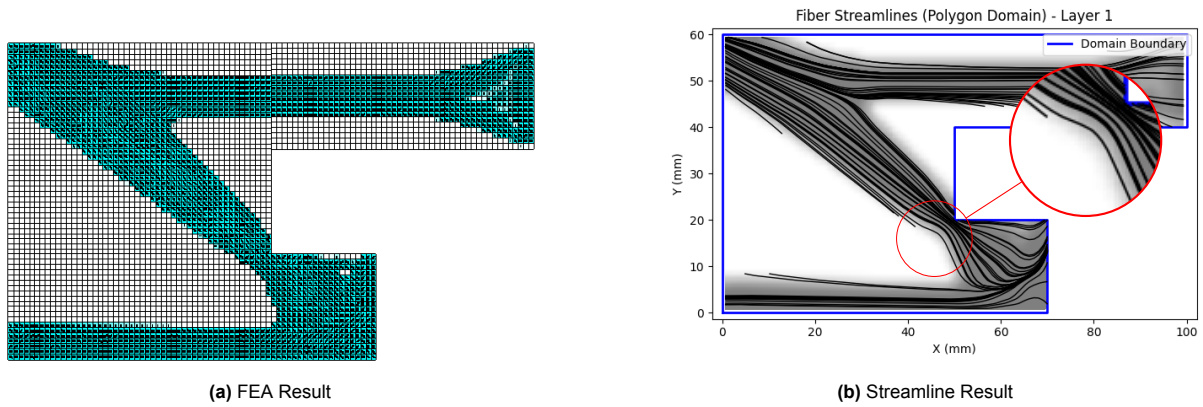


Figure 5.51: Mesh generation with element size 1mm

Table 5.19: Compliance change on element size

ElementSize	Compliance	Compliance Convergence	Iteration	CPU time
4.0	456.303622	$1e-6$	60	102s
2.0	320.049539	$1e-6$	59	646s
1.25	271.190823	$1e-6$	68	1625s
1.0	261.778082	$1e-6$	75	7211s

The results are as expected and show a gradual decrease in compliance and an exponential increase in CPU time. What the results also show is that for high stress areas like the corner the streamline has a somewhat higher curvature inside to limit the stress on that corner points. This becomes more visible with higher mesh resolutions as it can provide more direct specific fiber orientation changes. While an improvement in compliance and streamline can be achieved with a finer mesh the degree of improvement becomes in this case 3% for the compliance between 1mm and 1,25mm, but comes with an increase in CPU time for those cases of 343%. This shows that finer meshes than 1,25 for this case, while providing better results, is not favorable and very inefficient. The effect for different volume fractions and curvature constraint are also analysed and compared to their compliance.

Volume Fraction

Volume fraction analysis is performed on volume fractions $[0.3, 0.4, 0.5, 0.6]$ for the mesh of element size $2mm$. It is tested on the efficiency of the use of its material which is in this case defined by $MaterialEfficiency = \frac{1}{VolumeFraction * Compliance}$, and then compared to the base compliance result of volume fraction 1.0.

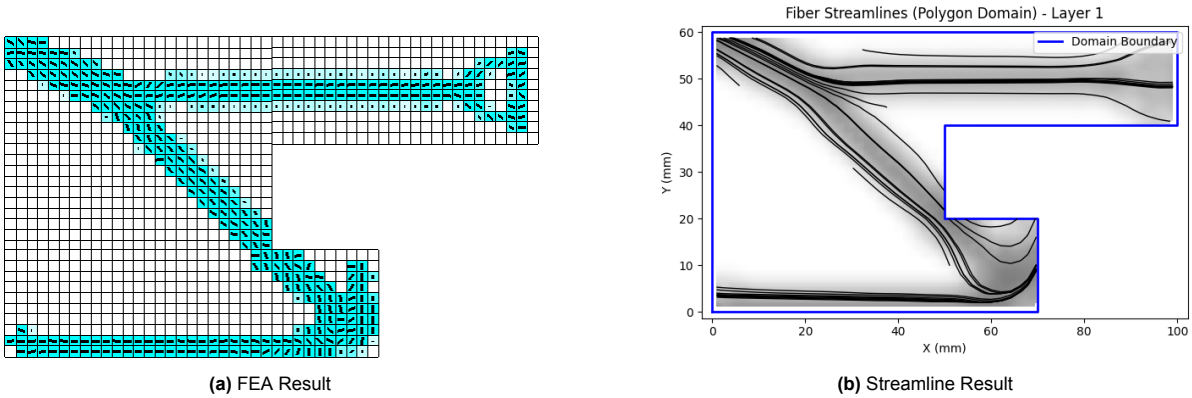


Figure 5.52: Volume Fraction of 0.3

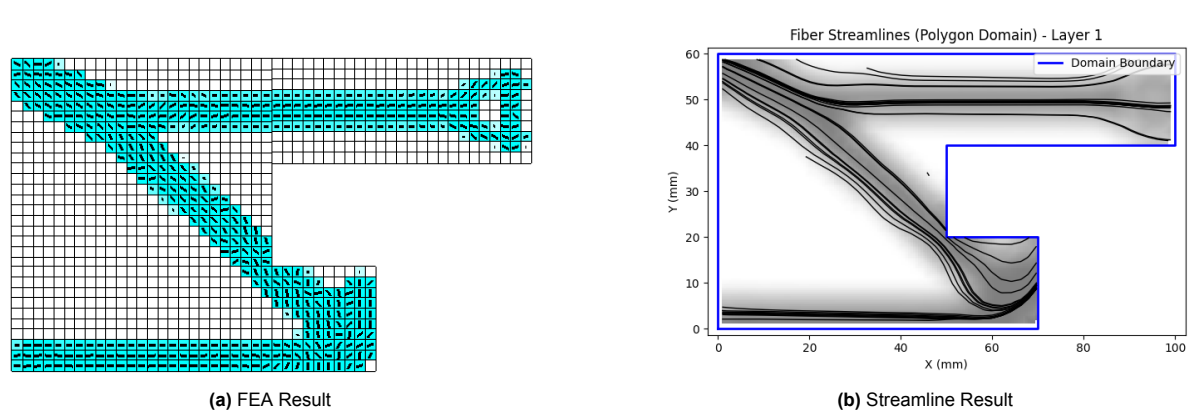


Figure 5.53: Volume Fraction of 0.4

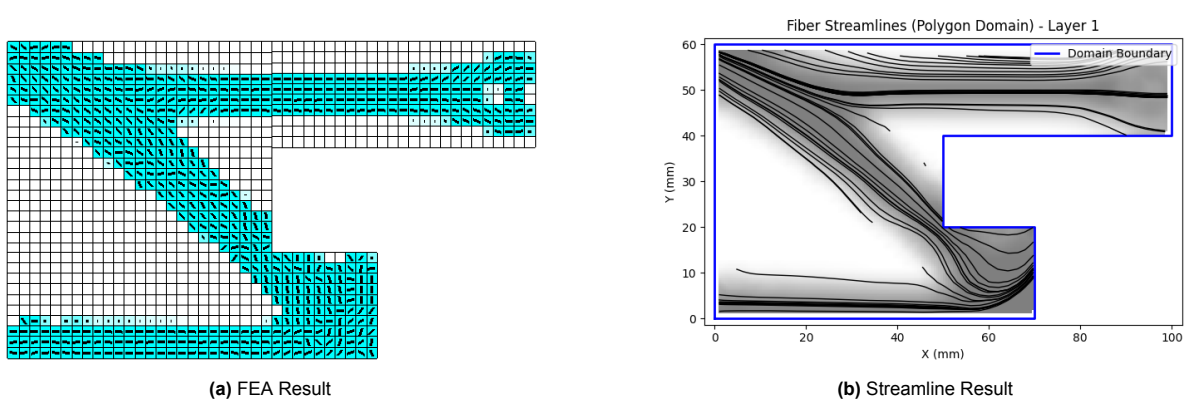


Figure 5.54: Volume Fraction of 0.5

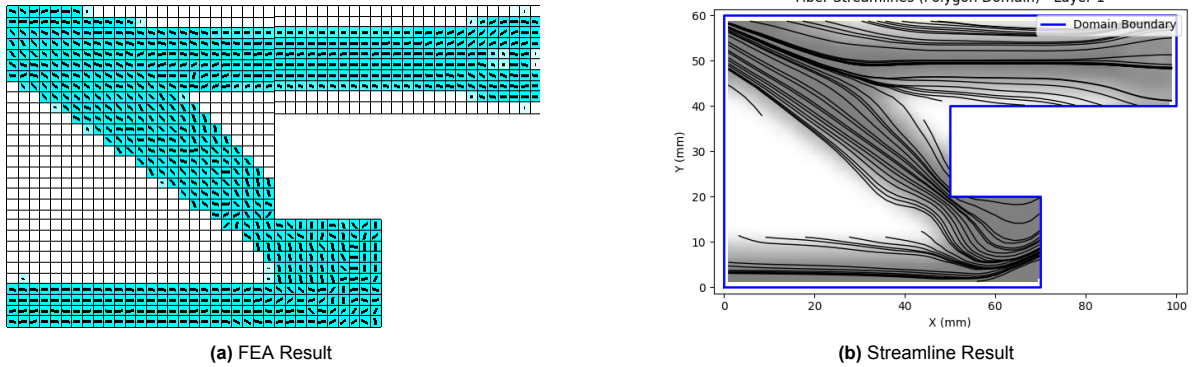


Figure 5.55: Volume Fraction of 0.6

Table 5.20: Compliance change on volume fraction

Volume Fraction	Compliance	Material Efficiency	Improvement Gain	Compliance Convergence	Iteration	CPU time
Base: 1.0	204.340314	0.0048938				
0.3	818.670561	0.0040716	-16.84%	$1e-6$	67	135s
0.4	451.904511	0.0055321	+13.04%	$1e-6$	72	179s
0.5	320.049539	0.0062490	+27.69%	$1e-6$	69	314s
0.6	266.066847	0.0062640	+28.00%	$1e-6$	85	1007s

The results show that around a volume fraction of 0.6 you have the best use of material in the design for the compliance, with an improvement of almost up to 30% of the base result. However, it also does show some increase in CPU time that comes with it that why a volume fraction of 0.5 with a negligible difference in material efficiency but a great decrease in CPU time is preferred for this case. This also shows that there is almost no benefit in trying to limit the design to a volume fraction of less than 0.4 as the efficiency of material for the gotten compliance is lacking more than the base value with more than -15% for the case of a 0.3 volume fraction. Next is an analysis on the curvature constraint for the problem is applied.

Curvature Constraint

The curvature constraint on the CBeam problem is applied for maximum curvature radii of $[2mm, 4mm, 5mm, 8mm, 10mm]$. To test these curvature constraints a volume fraction of 0.5 is used as it provided the best result in computational efficiency with material efficiency. The new material efficiency indexes are calculated for each curvature but against the base case of a zero curvature radius constraint.

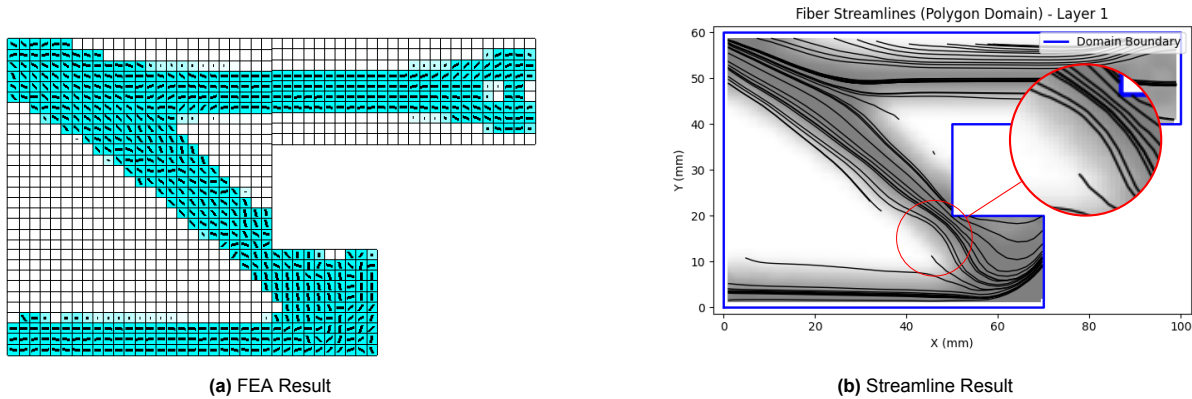


Figure 5.56: Curvature Constraint of 2mm

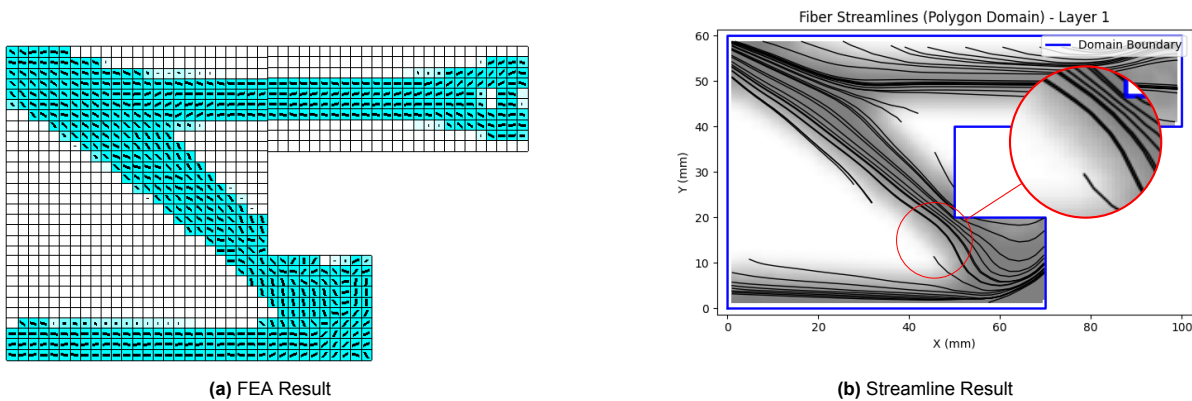


Figure 5.57: Curvature Constraint of 4mm

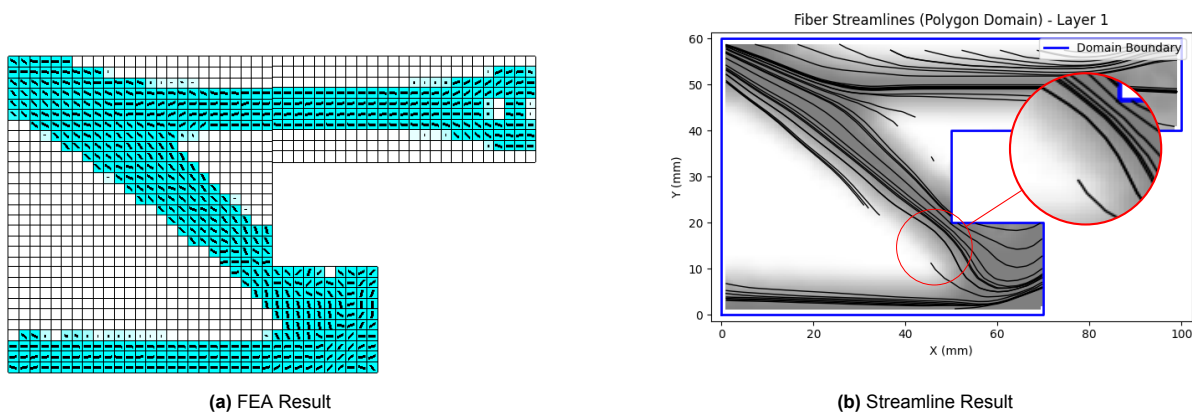


Figure 5.58: Curvature Constraint of 5mm

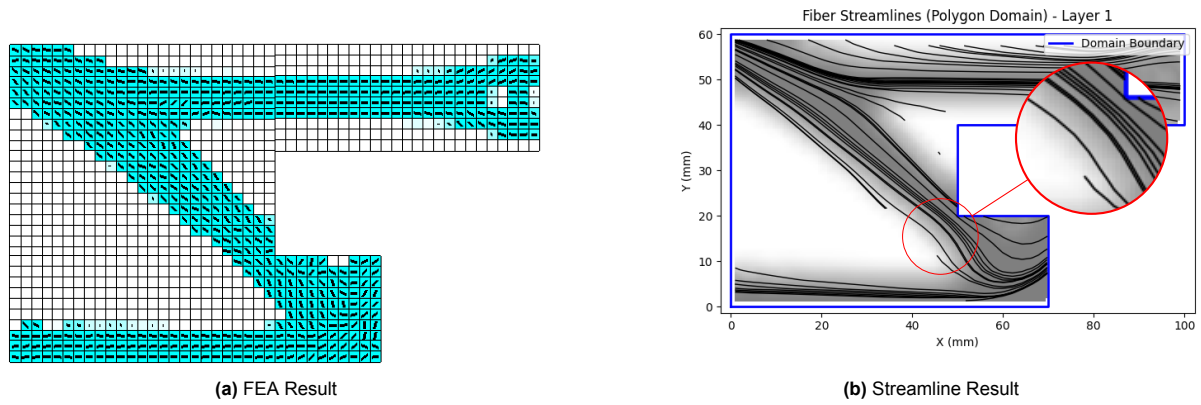


Figure 5.59: Curvature Constraint of 8mm

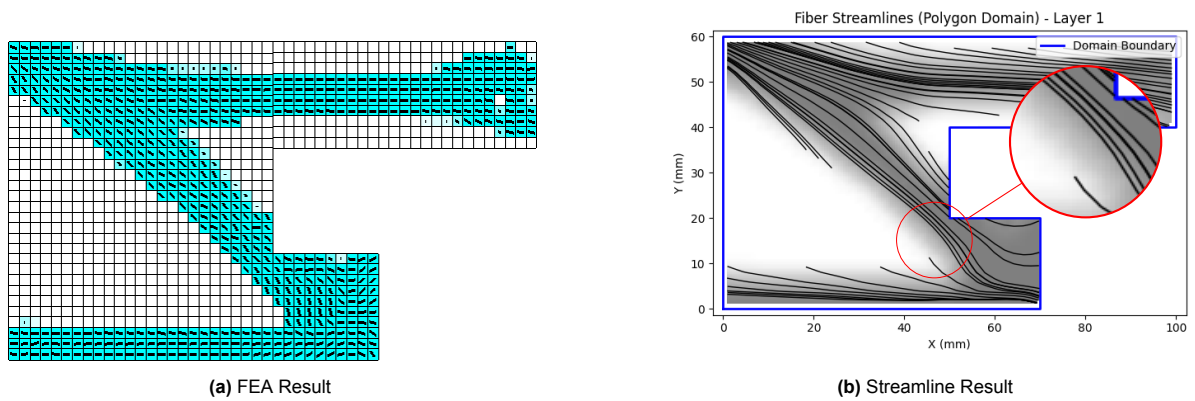


Figure 5.60: Curvature Constraint of 10mm

Table 5.21: Compliance change on curvature constraint

Curvature Constraint	Compliance	Material Efficiency	Improvement Gain	Compliance Convergence	Iteration	CPU time
Base: 0.05mm	314.214442	0.0063650				
2mm	320.049539	0.0062490	-1.82%	$1e-6$	59	646s
4mm	325.118186	0.0061516	-3.35%	$1e-6$	147	2390s
5mm	330.097823	0.0060588	-4.81%	$1e-6$	60	672s
8mm	328.728286	0.0060841	-4.41%	$1e-6$	70	1645s
10mm	367.584694	0.0054409	-14.52%	$1e-6$	133	2350s

The results of the curvature constraint analysis show that compared to the base case the stronger the curvature constraint the weaker the stiffness of the object as it has to follow these manufacturing limitations in the optimizations. What is also shown is that the material efficiency does not change a lot between the values off 4mm to 8mm probably as it shows the best compatibility with the stress paths followed throughout the object due to the forces. The 10mm constraint curvature case clearly shows the biggest problem with the compliance as it just shift out of that solution that can provide good load paths for the high stress corner area and instead by passe them with straight lines. Next is a comparative analysis applied on the difference between the topology and the fiber orientation optimization compared to a base result to see how the combined is compared to the separate.

Optimizers Comparative Analysis

In this section a comparative analysis on the optimizer separately and combined is taken. The analysis is follow the topology optimizer and the fiber optimizer independently and check the correlation of their results with the combined optimization result. There are a few difficulties in how the methodology works that intervene with a perfect comparative analysis. First is setting a base case for the comparison, however, the topology optimizer is limited by a volume fraction which means that a comparison to the case for a volume fraction of 1.0 is not comparable. The gamma penalty also has an effect on the initial density because if the density for the base case is set as 0.5, the compliance gets affected higher with an increased gamma due to the way the methodology works, however the case of a gamma of 1.0 can be used as base because if the final result is a true 0/1 region it will not be affected by the gamma and any result that has some amount of gray regions will either way show an increased compliance due to the gamma which shows an error but in the worse design direction. On top of that a similar problem is found in the fiber optimization where the standard penalty penalization has a higher value in the initial state then in the final optimized state which is not a good comparison as it means a more distributed material property, so to test this different penalties are taken and the rate of change of the compliance in comparison is taken which is defined as $100 * \frac{MateEff^{p-1} - MateEff^p}{P_g^{p-1} - P_g^p}$. Different initial points were tested and shown below in the table:

Table 5.22: Compliance change rate on penalty parameters

Volume Fraction	Penalty Standard Dev	Penalty Density Power	Compliance	Material Efficiency	Change Rate
1.0	4.0	1.0	279.716668	0.003575	(-)
1.0	1.0	1.0	331.434731	0.003017	0.0186
1.0	0.5	1.0	409.616010	0.002441	0.1152
1.0	0.25	1.0	456.021133	0.002193	0.0992
1.0	0.125	1.0	456.197917	0.002192	0.0008
0.5	4.0	1.0	559.433335	0.003575	(-)
0.5	1.0	1.0	662.869462	0.003017	0.0186
0.5	0.5	1.0	819.232019	0.002441	0.1152
0.5	0.25	1.0	912.042265	0.002193	0.0992
0.5	0.125	1.0	912.395834	0.002192	0.0008
0.5	0.125	3.0	3649.583335	0.00002740	(-)

These results show that even though the volume fraction has an effect on the base compliance, it does not have any effect on the material efficiency of the base case as expected. What is also shown is that the rate of change of compliance reduces with an increasing penalty parameter. This also shows that after the penalty reaches 0.25 to 0.125 the rate converges. What should also be shown is the penalty with which the compliance is increased for a base case with density penalty parameter of 3.0, it shows that the weight of not using a strict 1/0 result on the compliance. Thus, the Base Case, which is compared against, is taken with the volume fraction of 0.5, standard deviation penalty parameter off 0.125, and compared for both density penalty parameter 1.0 and 3.0. The tests are done on a curvature constraint of 4mm as it showed some good averaging result together with the 5mm and 8mm case, but with a better CPU time. The results of these tests are shown below

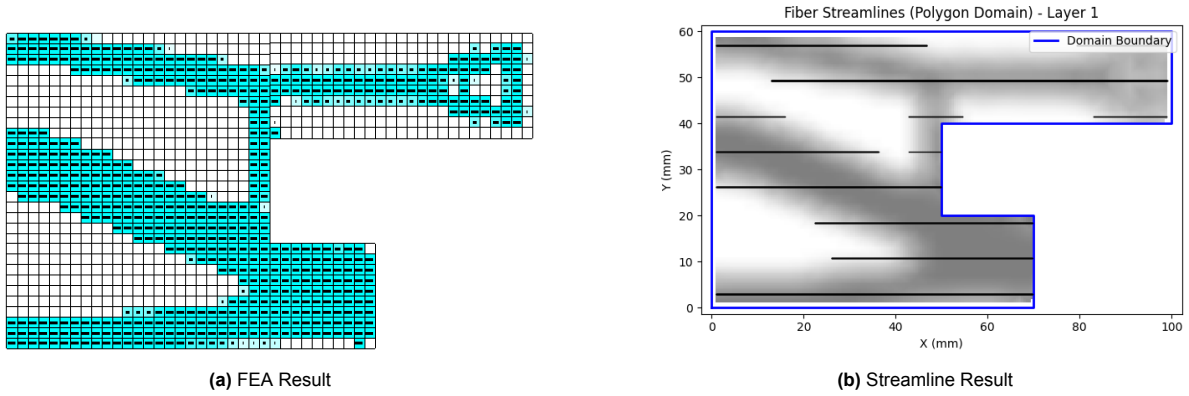


Figure 5.61: Topology Optimization

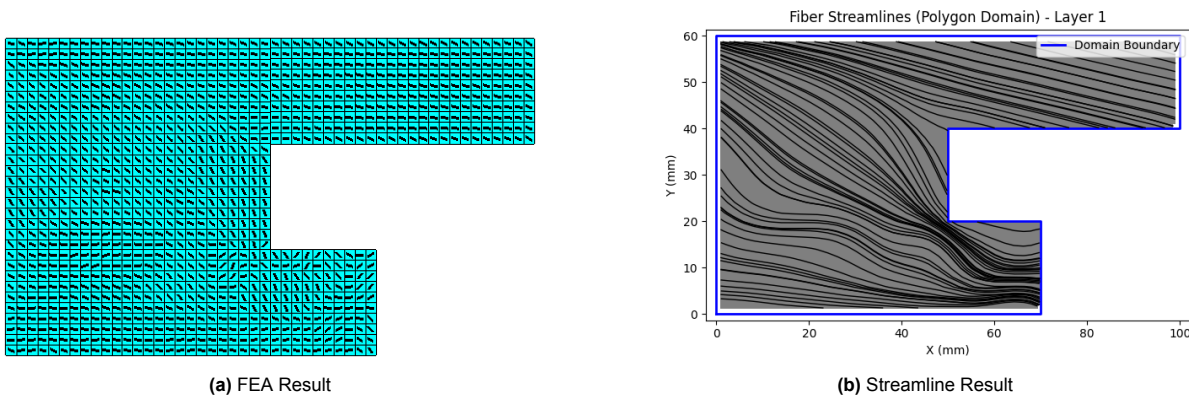


Figure 5.62: Fiber Optimization

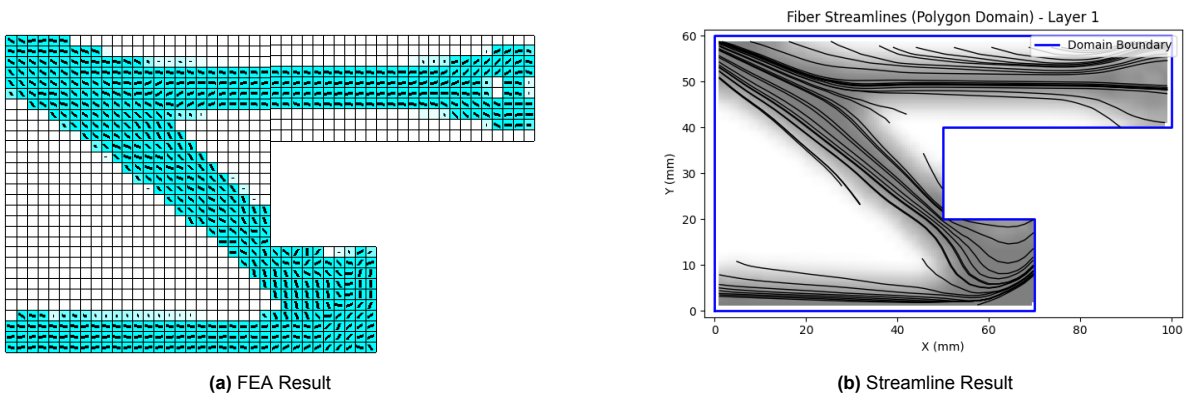


Figure 5.63: Combined Optimization

The results show some interesting thing, it shows that the topology optimizer is the most drastic participant in decreasing the compliance to a level of almost 4x the base case. The fiber optimizer also shows that there is no difference in the compliance change if the density penalty parameter is 1.0 or 3.0. The improvements by the fiber optimization is also around 2x the base case. In base case thinking if the topology optimizer can improve the base case by 3.77x and the fiber optimizer can improve it by 2.13x then it would be expected that the combined result would be improved by 8.03x, however the result show an improvement of 11.23x which shows that the topology optimization and fiber optimization come hand in hand for an amplifying improvement greater than each separate improvement.

Table 5.23: Comparative compliance analysis

Optimization	Compliance	Material Efficiency	Improvement Gain 1.0	Improvement Gain 3.0
Base 1.0	912.395834	0.002192	(-)	(-)
Base 3.0	3649.583335	0.0005480	(-)	(-)
Topology	968.124812	0.002066	(-)	+377.01%
Fiber 1.0	427.807525	0.004675	+213.28%	(-)
Fiber 3.0	1711.23009	0.00116875	(-)	+213.28%
Combined	325.118186	0.006152	(-)	+1122.63%

Conclusion and Recommendations

Composite laminate optimizations are being used with many methods and strategies developed in the past decades. Grown by the innovation of additive manufacturing, the composite industry of today allows to follow more the optimal designs found by the many optimizers around. However, the optimization methods of composites are very complex and expensive as they offer another degree of freedom compared to isotropic topology optimizations. Many of these methods become unusable or theoretical optimals that are unable to be physically manufactured. This thesis, a model focused on its manufacturability for these composites is developed.

In topology optimization the SIMP method provides an easy way to mathematically express the application of material in the design domain, which combined with a GCMMA solves for the optimal topology fast. For the Fiber optimization the ANDFO shows a clear improved adaptive strategy to decent the fiber angles into their optimal position by using normal distributions and adaptive strategies, it's non-convex nature is limited through the use of the GCMMA solver by staying on a global convergence route to the solution. In this thesis an approach to combining both these methods with a focus on the manufacturability of the result is taken where a mix of heuristic and concurrent methods are applied.

Verifications were applied using different literary sourced problems to test the validity of the model. Continuing with a parametric analysis on the different parameters of the system ranging from the filter radii to power penalties and more. Following the parametric analysis some tests are run to test the full scope of the system, a random initialization point because the problem is so non-convex it stands to reason a different starting setup could result in a different result of the system. Finally, a CBeam is taken as main test to check the model's handling of sharp corners cutting through the stress paths and its resulting shapes.

In regards to the verifications, it is clear that the model follows the expected results from the literature for the most part. The topologies are quite similar to the topologies resulting from the problems in the studies, however, the model often shows stricter results with the GCMMA solver applied on the SIMP method than in the literature as expected due to the fact that the GCMMA solver optimizes the problem faster towards its penalized discrete 0 1 result. The VSO optimizer verification is harder to verify as the similarities are more complex to check. However, the results do show that the fiber orientation paths follow somewhat similar path lines for the stress to travel through as the results from the literature. As for the combined verification, the result shows great similarity for the second test that was applied both in fiber placement and in topology. The first test however while the fibers do show somewhat similar paths and the topology as well, there is a slight difference in the lower beam of the topology where some of the load is transferred differently. This is likely due to the difference in topology optimization method which for the non convexity of the problem can result in a different topology depending on whether one uses the SIMP method or the Heaviside method.

As for the parametric analysis, it shows that the filter parameters have an influential effect on the final result of the composite design and its compliance. The topology filter for both the combined tests and the separate tests show that the value at which checker boarding is removed is a minimum of twice the

element's size, this is to allow the elements density variable to be affected enough by its neighbors to create this dependency on them. However, in the combined result there are still some signs of checker boarding formed when using twice the element's size suggesting that for a full removal in future models a radius of three times the element's size is preferred. As for fiber filter a different effect is seen, the known checker boarding pattern for fiber orientation occurs if the fiber filter is too small which in the tests is shown to occur for radii below 3 times the element's size. On top of that a somewhat higher degree of path following is seen for radii above 5 times the element's size as it provides a more global direction per truss element. The power on the filter of the fiber is also shown to have some improved effect when at the second power, as it uses the global direction of its neighbors but with a higher dependency of the neighbors closer to itself as well as improved results on junctions points.

The random starting variables also provided some interesting results, it showed that the topology optimization in the combined model is only slightly affected in its secondary trusses in the final design pointing to the fact that the topology optimizer will somewhat always result in a similar form showing that it has only a low level of non-convexity compared to the fiber optimization. The fiber optimization has a much higher sensitivity to the starting variable values because it has a direct effect on the stiffness direction and with a higher non-convexity to the problem it falls more into local minima with these starting conditions, however an increase in the scale at which the curvature constraint is applied in the GCMMA model does show improved results for the fiber optimizations as they enforce this constraint more, however, at the price of higher CPU time.

The CBeam design test also shows the effects of the model on different mesh generation levels. The model follows a similar geometry for all 4 of the generated mesh sizes, but when the element size is too small the resulting geometry sometimes results in weird choices for material where certain spots can be formed along a truss showing the need for material on that truss while being limited. The fiber on all 4 meshes also show similar paths and with higher meshes they show good performance in working around topology holes. However on higher stress location these fibers can still show some deviating results for fiber orientation paths, but can follow the curvature constraints in more detail than at thicker meshes. Also the volume fraction and curvature fractions were tested on the CBeam problem and showed that certain levels of volume fraction allow for the most efficient use of material which is decreased from the full design space and if too low a volume fraction is required. The curvature constraint tests showed that while the problem had to adhere to certain curvatures there was a certain go to curvature the problem fell into that showed the greatest stagnation in compliance and similar streamline curved results around the high stress area. The optimizer compared analysis also showed the need for these combined optimizations as they amplify the improvement to the compliance of each of their optimization more than the optimizers separately do.

Generally, the model shows great promise towards manufacturability with its improved fiber orientation paths around corners or truss splits. The model also has a good discrete topology modeling that helps with the manufacturability of the design. And the addition of the curvature constraint help with covering the bridge that certain machine limitations impose.

6.1. Recommendations

The system developed in this thesis still has some room for improvement, different theories and methods can be applied to the code to help improve the manufacturability even more. On top of that the out of plane effect with bending and multi-layer testing can be applied with different mathematical models. The model can also still improve on the computational time as there is much room for improvement in streamlining and minimizing computations using caches for pre-learned data and just more efficient front-end coding in general. The model at the moment also doesn't utilize the full potential of a Gauss integration and still has a focus on an elemental composition of the laminate. Which can be done through nodal variables which provides also a change in angle over the element, however, this increase in accuracy and physical correctness comes with also a drastic increase in complexity as the constitutive matrix would have to be interpolated at every gauss point as well as the sensitivity which would to be recalculated into each node affected by each element separately.

Bibliography

- [1] Mazen A. Albazzan et al. "Efficient design optimization of nonconventional laminated composites using lamination parameters: A state of the art". In: *Composite Structures* 209 (2019), pp. 362–374. ISSN: 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2018.10.095>. URL: <https://www.sciencedirect.com/science/article/pii/S0263822318321664>.
- [2] José Humberto S. Almeida Jr. et al. "A concurrent fibre orientation and topology optimisation framework for 3D-printed fibre-reinforced composites". In: *Composites Science and Technology* 232 (2023), p. 109872. ISSN: 0266-3538. DOI: <https://doi.org/10.1016/j.compscitech.2022.109872>. URL: <https://www.sciencedirect.com/science/article/pii/S0266353822006145>.
- [3] Haichao An, Shenyang Chen, and Hai Huang. "Laminate stacking sequence optimization with strength constraints using two-level approximations and adaptive genetic algorithm". In: *Structural and Multidisciplinary Optimization* 51.6 (2015), pp. 1261–1276. DOI: 10.1007/s00158-014-1181-0. URL: <https://doi.org/10.1007/s00158-014-1181-0>.
- [4] M. Autio. "Determining the real lay-up of a laminate corresponding to optimal lamination parameters by genetic search". In: *Structural and Multidisciplinary Optimization* 20.4 (2000), pp. 301–313. DOI: 10.1007/s001580050160. URL: <https://doi.org/10.1007/s001580050160>.
- [5] M.P. Bendsøe. "Optimal shape design as a material distribution problem." In: *Structural Optimization* 1 (1989), pp. 193–202. URL: <https://doi.org/10.1007/BF01650949>.
- [6] Martin P. Bendsøe and Ole Sigmund. "Material interpolation schemes in topology optimization". In: *Archive of Applied Mechanics* 69.9-10 (1999), pp. 635–654. DOI: 10.1007/s004190050248.
- [7] S Bulman, J Sienz, and E Hinton. "Comparisons between algorithms for structural topology optimization using a series of benchmark studies". In: *Computers & Structures* 79.12 (2001), pp. 1203–1218. ISSN: 0045-7949. DOI: [https://doi.org/10.1016/S0045-7949\(01\)00012-8](https://doi.org/10.1016/S0045-7949(01)00012-8). URL: <https://www.sciencedirect.com/science/article/pii/S0045794901000128>.
- [8] Eralp Demir, Pouya Yousefi-Louyeh, and Mehmet Yildiz. "Design of variable stiffness composite structures using lamination parameters with fiber steering constraint". In: *Composites Part B: Engineering* 165 (2019), pp. 733–746. ISSN: 1359-8368. DOI: <https://doi.org/10.1016/j.compositesb.2019.02.004>. URL: <https://www.sciencedirect.com/science/article/pii/S1359836818336631>.
- [9] Zunyi Duan et al. "Concurrent multi-material and multi-scale design optimization of fiber-reinforced composite material and structures for minimum structural compliance". In: *Composite Structures* 311 (2023), p. 116796. ISSN: 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2023.116796>. URL: <https://www.sciencedirect.com/science/article/pii/S026382232300140X>.
- [10] Zunyi Duan et al. "Concurrent multi-scale design optimization of fiber-reinforced composite material based on an adaptive normal distribution fiber optimization scheme for minimum structural compliance and additive manufacturing". In: *Computer Methods in Applied Mechanics and Engineering* 434 (2025), p. 117596. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2024.117596>. URL: <https://www.sciencedirect.com/science/article/pii/S0045782524008508>.
- [11] Hisao Fukunaga and Hideki Sekine. "A Laminate Design for Elastic Properties of Symmetric Laminates with Extension-Shear or Bending-Twisting Coupling". In: *Journal of Composite Materials* 28.8 (May 1994), pp. 708–731. DOI: 10.1177/00219983940280080. URL: <https://doi.org/10.1177/00219983940280080>.

- [12] Xueyan Hu et al. "An adaptive method of moving asymptotes for topology optimization based on the trust region". In: *Computer Methods in Applied Mechanics and Engineering* 393 (2022), p. 114202. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2021.114202>. URL: <https://www.sciencedirect.com/science/article/pii/S0045782521005338>.
- [13] Delin Jiang, Robert Hoglund, and Douglas E. Smith. "Continuous Fiber Angle Topology Optimization for Polymer Composite Deposition Additive Manufacturing Applications". In: *Fibers* 7.2 (2019). DOI: 10.3390/fib7020014. URL: <https://www.mdpi.com/2079-6439/7/2/14>.
- [14] C.Y. Kiyono, E.C.N. Silva, and J.N. Reddy. "A novel fiber optimization method based on normal distribution function with continuously varying fiber path". In: *Composite Structures* 160 (2017), pp. 503–515. ISSN: 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2016.10.064>. URL: <https://www.sciencedirect.com/science/article/pii/S0263822316314957>.
- [15] C.Y. Kiyono, E.C.N. Silva, and J.N. Reddy. "A novel fiber optimization method based on normal distribution function with continuously varying fiber path". In: *Composite Structures* 160 (2017), pp. 503–515. ISSN: 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2016.10.064>. URL: <https://www.sciencedirect.com/science/article/pii/S0263822316314957>.
- [16] Jaewook Lee et al. "Topology optimization for continuous and discrete orientation design of functionally graded fiber-reinforced composite structures". In: *Composite Structures* 201 (2018), pp. 217–233. ISSN: 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2018.06.020>. URL: <https://www.sciencedirect.com/science/article/pii/S0263822318305798>.
- [17] Yi LIU et al. "Variable stiffness design optimization of fiber-reinforced composite laminates with regular and irregular holes considering fiber continuity for additive manufacturing". In: *Chinese Journal of Aeronautics* 38.3 (2025), p. 103317. ISSN: 1000-9361. DOI: <https://doi.org/10.1016/j.cja.2024.11.027>. URL: <https://www.sciencedirect.com/science/article/pii/S1000936124004710>.
- [18] Zebei Mao et al. "An equivalent anisotropy orientation tensor algorithm for integrated material-structure design". In: *Computer Methods in Applied Mechanics and Engineering* 420 (2024), p. 116720. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2023.116720>. URL: <https://www.sciencedirect.com/science/article/pii/S0045782523008435>.
- [19] Ali Odeh et al. "Analysis of Laminated Composite Plates: A Comprehensive Bibliometric Review". In: *Buildings* 14.6 (2024). ISSN: 2075-5309. DOI: 10.3390/buildings14061574. URL: <https://www.mdpi.com/2075-5309/14/6/1574>.
- [20] OpenPR. *Advanced Composites Market Predicted to Reach US\$71.0 Billion by 2033, Fueled by EV Lightweighting and Aviation Fleet Renewal*. Accessed: 2026-02-19. Market Research Insights. 2024. URL: <https://www.openpr.com/news/4393315/advanced-composites-market-predicted-to-reach-us-71-0-billion> (visited on 02/19/2026).
- [21] Nikoleta Pasvanti. "Topology optimization, variable stiffness design and staggered optimization for laminated composite structures: A comparative study". Master's thesis. Delft, Netherlands: Delft University of Technology, 2022. URL: <https://repository.tudelft.nl/>.
- [22] Daniel Peeters. "Design Optimisation of Practical Variable Stiffness and Thickness Laminates". Doctoral thesis. Delft, Netherlands: Delft University of Technology, 2017. DOI: 10.4233/uuid:a07ea6a4-be73-42a6-89b5-e92d99bb6256. URL: <https://doi.org/10.4233/uuid:a07ea6a4-be73-42a6-89b5-e92d99bb6256>.
- [23] Zheng Qiu et al. "Concurrent topology and fiber orientation optimization method for fiber-reinforced composites based on composite additive manufacturing". In: *Computer Methods in Applied Mechanics and Engineering* 395 (2022), p. 114962. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2022.114962>. URL: <https://www.sciencedirect.com/science/article/pii/S0045782522002213>.
- [24] Ole Sigmund. "A 99 line topology optimization code written in Matlab". In: *Structural and Multidisciplinary Optimization* 21.2 (2001), pp. 120–127. DOI: 10.1007/s001580050176. URL: <https://doi.org/10.1007/s001580050176>.
- [25] Jian Song et al. "Topology Optimization-Driven Design for Offshore Composite Wind Turbine Blades". In: *Journal of Marine Science and Engineering* 10.10 (2022). ISSN: 2077-1312. DOI: 10.3390/jmse10101487. URL: <https://www.mdpi.com/2077-1312/10/10/1487>.

- [26] Yunlong Tang, Aidan Kurtz, and Yaoyao Fiona Zhao. "Bidirectional Evolutionary Structural Optimization (BESO) based design method for lattice structure to be fabricated by additive manufacturing". In: *Computer-Aided Design* 69 (2015), pp. 91–101. ISSN: 0010-4485. DOI: <https://doi.org/10.1016/j.cad.2015.06.001>. URL: <https://www.sciencedirect.com/science/article/pii/S0010448515000792>.
- [27] Julien M.J.F. Van Campen, Christos Kassapoglou, and Zafer Gürdal. "Design of fiber-steered variable-stiffness laminates based on a given lamination parameters distribution". English. In: *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. 52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference ; Conference date: 04-04-2011 Through 07-04-2011. United States: American Institute of Aeronautics and Astronautics Inc. (AIAA), Dec. 2011. DOI: 10.2514/6.2011-1894.
- [28] Georges Verchery. "Design rules for the laminate stiffness". In: *Mechanics of Composite Materials* 47.1 (Mar. 2011), pp. 47–58. DOI: 10.1007/s11029-011-9186-x. URL: <https://doi.org/10.1007/s11029-011-9186-x>.
- [29] Michael Yu Wang, Xiaoming Wang, and Dongming Guo. "A level set method for structural topology optimization". In: *Computer Methods in Applied Mechanics and Engineering* 192.1 (2003), pp. 227–246. ISSN: 0045-7825. DOI: [https://doi.org/10.1016/S0045-7825\(02\)00559-5](https://doi.org/10.1016/S0045-7825(02)00559-5). URL: <https://www.sciencedirect.com/science/article/pii/S0045782502005595>.
- [30] ZhenZhou Wang and Adam Sobey. "A comparative review between Genetic Algorithm use in composite optimisation and the state-of-the-art in evolutionary computation". In: *Composite Structures* 233 (2020), p. 111739. ISSN: 0263-8223. DOI: <https://doi.org/10.1016/j.compstruct.2019.111739>. URL: <https://www.sciencedirect.com/science/article/pii/S0263822319328636>.
- [31] Zhangming Wu, Gangadharan Raju, and Paul M. Weaver. "Framework for the Buckling Optimization of Variable-Angle Tow Composite Plates". In: *AIAA Journal* 53.12 (2015), pp. 3788–3804. DOI: 10.2514/1.J054029. URL: <https://doi.org/10.2514/1.J054029>.
- [32] Furong Xie et al. "Topology optimization for fiber-reinforced plastic (FRP) composite for frequency responses". In: *Computer Methods in Applied Mechanics and Engineering* 428 (2024), p. 117114. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2024.117114>. URL: <https://www.sciencedirect.com/science/article/pii/S0045782524003700>.
- [33] Yanan Xu et al. "Concurrent optimization of topological configuration and continuous fiber path for composite structures — A unified level set approach". In: *Computer Methods in Applied Mechanics and Engineering* 399 (2022), p. 115350. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2022.115350>. URL: <https://www.sciencedirect.com/science/article/pii/S0045782522004340>.
- [34] Xiaolei Yan et al. "Concurrent optimization of macrostructures and material microstructures and orientations for maximizing natural frequency". In: *Engineering Structures* 209 (2020), p. 109997. ISSN: 0141-0296. DOI: <https://doi.org/10.1016/j.engstruct.2019.109997>. URL: <https://www.sciencedirect.com/science/article/pii/S0141029619337551>.
- [35] Xuyu Zhang et al. "A review of structural topology optimization for fiber-reinforced composites". In: *Composites Part B: Engineering* 299 (2025), p. 112393. ISSN: 1359-8368. DOI: <https://doi.org/10.1016/j.compositesb.2025.112393>. URL: <https://www.sciencedirect.com/science/article/pii/S1359836825002859>.