Optimization of Composite Structures with nonlinear geometric behavior using Memetic Algorithms supported by learning procedures

Ricardo Oliveira de Faria

Master's dissertation

Supervisor: Carlos Alberto da Conceição António

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Master in Mechanical Engineering

July 2024

Abstract

A multi-objective optimization framework for the sizing and anisotropic topology of geometrically-nonlinear beam-stiffened shell structures made of fiber-reinforced plastics is proposed. Minimum weight, strain energy and strain energy variability are the objectives of the proposed robust design approach and their trade-off is optimized while adhering to stress, displacement and buckling constraints. The design variables are the ply angles and thicknesses of shell laminates, the stiffeners' cross-sectional dimensions and those related to material selection and distribution at both the laminate and structure levels.

The proposed Multi-Objective Memetic Algorithm (MOMA) exploits the synergy between multiple Lamarckian and Baldwinian learning procedures. Design space decomposition is achieved through the use of multiple sequentially-evolving subpopulations (SPs), while a virtual population (VP) with age-dominance dual nature updates the Pareto front and accelerates the global search. The concept of *species* serves as an abstraction of the search space discontinuities due to the presence of integer-type design variables.

The finite element model for structural analysis was developed from a pre-existing Fortran program. Its extensive adaptation resulted in a reduced execution time, both through the optimization of the program itself while following more recent language standards and, on the other hand, the total rewrite of the program's interface, to make it an integral part of the optimization algorithm (OA).

The OA was generalized for an arbitrary number of objectives. Dominance evaluation was extended to offspring candidates at the operator level, with the option of including the solutions from either the SPs or the VP in that evaluation. An evolution history was added which enables every solution generated throughout the evolutionary process to be stored. An interface for the automatic generation of graphs from the results of the OA was developed in Python. The MOMA was validated through its application to four multi-objective optimization problems.

Keywords memetic · multi-objective · hybrid composites · geometric nonlinearity · parallelism

Resumo

Uma abordagem de otimização multiobjetivo para o dimensionamento e topologia anisotrópica de estruturas geometricamente não-lineares feitas a partir de plásticos reforçados por fibras é proposta. O compromisso entre os objetivos minimização do peso, energia de deformação e variabilidade da energia é otimizado obedecendo a restrições de tensão, deslocamento e encurvadura. As variáveis de projeto são os ângulos e espessuras das camadas dos laminados de casca, as dimensões das secções transversais das vigas e a seleção e distribuição dos materiais ao nível dos laminados e da estrutura.

O Algoritmo Memético Multiobjetivo (MOMA) proposto explora a sinergia entre diversos processos de aprendizagem Lamarckianos e Baldwinianos. A decomposição do espaço de pesquisa é efetuada através do uso de múltiplas subpopulações (SPs) que evoluem de forma sequencial, enquanto uma população virtual (VP) com natureza dupla de idade-dominância atualiza a frente de Pareto e acelera a pesquisa global. O conceito de *espécie* é usado como abstração para as descontinuidades devidas à presença de variáveis de projeto inteiras.

O modelo de elementos finitos foi desenvolvido a partir de um programa em Fortran preexistente. A sua extensiva adaptação permitiu reduzir o tempo de execução, através da otimização do próprio programa, acompanhada da implementação dos *standards* mais recentes da linguagem, bem como da reescrita completa da sua interface, de forma a torná-lo uma parte integral do algoritmo de otimização (OA).

O OA foi generalizado para um número arbitrário de objetivos. A avaliação da dominância foi estendida aos descendentes candidatos ao nível dos operadores, com a opção de incluir soluções quer das SPs quer da VP nessa avaliação. Foi acrescentado um histórico de evolução que permite registar todas as soluções geradas pelo programa. Uma interface para a geração automática de gráficos a partir dos resultados do OA foi desenvolvida em Python. O MOMA foi validado através da sua aplicação a quatro problemas multiobjetivo.

Palavras-chave memético · multiobjetivo · compósitos híbridos · geometria não-linear · paralelismo

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Carlos António, for his invaluable mentoring and ever-present support throughout the development of this project. His expertise was instrumental in the successful undertaking of this research topic. Moreover, his patience and continued encouragement were very positive to the outcome of this thesis.

I am forever grateful to my family, and especially my parents, for always supporting me in attaining my academic goals.

To my dearest friends, I express considerable appreciation for their companionship and their demonstrated interest in my academic journey. I especially thank my lifelong friend David for being part of this journey, and alongside whom I've made a substantial amount of my gym progress, which is so important for a balanced and healthy life.

Contents

Abbreviations **xi**

List of Algorithms **xiii**

List of Figures **xvi**

List of Symbols **xvii**

List of Tables **xix**

Listings **xxi**

1 Introduction **1**

1.1 Objectives 2

1.2 Organization 2

2 State of the Art **5**

2.1 Genetic & Memetic algorithms: a synergistic view 5

2.2 Lamarckian vs Baldwinian learning 6

2.3 Generations of Memetic Algorithms 7

2.4 Recent developments in optimization of FRP structures 8

3 Building the optimization model for hybrid composite structures **9**

3.1 Three Columns approach to a MOOP 10

3.2 Model of the system: Finite element model 12

3.2.1 Finite element formulation 12

3.2.2 Finite element discretization 15

3.2.3 Solution method 17

3.2.4 Structural integrity analysis 20

3.2.5 Approximation of the strain energy gradient . . . 23
3.3 Optimization model for the MOOP . . . 24
3.3.1 Objective functions for the MOOP . . . 25
3.3.2 MOOP formulations (case studies) . . . 26
3.3.3 Review of concepts in multi-objective optimization . . . 27
3.4 Optimization algorithm: Memetic algorithm . . . 29

4 Multi-Objective Memetic Algorithm 33
4.1 MOMA framework . . . 33
4.2 Design space decomposition: Subpopulations (SP) . . . 35
4.2.1 Local fitness assignment . . . 36
4.3 Evolution history: the Virtual Population (VP) . . . 37
4.4 Memes . . . 39
4.5 Mutation operators . . . 41
4.5.1 Implicit Mutation . . . 41
4.5.2 Controlled Mutation . . . 41
4.6 Crossover operators . . . 42
4.6.1 Elitist parameterized Uniform Crossover (EpUC) . . . 42
4.6.2 Elitist hybrid crossover with genetic improvement (EHCgi) . . . 43
4.6.3 Age-Dominance parameterized Uniform crossover (ADpUC) . . . 45
4.6.4 Age-Dominance parameterized Selfish Gene crossover (ADpSG) . . . 46
4.7 Similarity control and evolution history . . . 47

5 Improving the efficiency of the Finite Element program 49
5.1 General aspects of the original algorithm . . . 49
5.1.1 Read/write operations . . . 49
5.2 Requirements for parallelism . . . 50
5.3 The FEM solver as part of the optimization algorithm . . . 51

6 Applications 53
6.1 Case Study 1: Weight vs Energy minimization . . . 56
6.2 Case Study 2: Weight vs Energy variability minimization . . . 58
6.3 Case Study 3: Tri-objective optimization problem . . . 59
6.4 Case Study 4: an iteration of the original 3OOP . . . 60

7 Conclusions 65
7.1 Future works . . . 66

Abbreviations

2OOP	Bi-Objective Optimization Problem
3OOP	Tri-Objective Optimization Problem
API	Application Programming Interface
CFRP	Carbon Fiber-Reinforced Plastic
EA	Evolutionary Algorithm
FEM	Finite Element Method
FPF	First Ply Failure
FRP	Fiber-Reinforced Plastic
GA	Genetic Algorithm
GFRP	Glass Fiber-Reinforced Plastic
GPU	Graphics Processing Unit
HPC	High-Performance Computing
KFRP	Kevlar Fiber-Reinforced Plastic
MA	Memetic Algorithm
MOMA	Multi-Objective Memetic Algorithm
MOOP	Multi-Objective Optimization Problem
OA	Optimization Algorithm
RDO	Robust Design Optimization
RGB	Red, Green, Blue
SP	Subpopulation
VP	Virtual Population

List of Algorithms

1 Multi-Objective Memetic Algorithm (MOMA) 34
2 Implicit Mutation 41
3 Controlled Mutation 41
4 Uniform Crossover 42
5 Elitist parameterized Uniform Crossover (EpUC) 42
6 Elitist Hybrid Crossover with genetic improvement (EHCgi) 44
7 Age-Dominance parameterized Uniform Crossover (ADpUC) 46
8 Age-Dominance parameterized Selfish Gene crossover (ADpSG) 47

List of Figures

3.1 Framework for the design of hybrid composite structures 10
3.2 Three columns approach to an optimization problem 11
3.3 Beam-reinforced shallow shell structure [1] 12
3.4 Definition of design variables for sizing and topological parts for hybrid composite structures [2] 25
3.5 Type collaboration graph 30

4.1 Proposed structure for the Multi-Objective Memetic Algorithm (MOMA) . . . 34
4.2 Fitness assignment and local evolutionary scheme during the isolation stage . . 38

6.1 Spherical shell structure and laminate definition. The beam laminate stiffeners are displayed in bold [2]. 53
6.2 Boundary conditions for $\frac{1}{4}$ of the spherical shell composite structure. Nodes that are colored blue have prescribed displacements and rotations, black nodes represent hinges. A point load is applied to the center node (0,0). 54
6.3 Pareto front - 1^{st} case study 57
6.4 Evolution of the Pareto front throughout the epochs 1^{st} case study 58
6.5 Pareto front - 2^{nd} case study 59
6.6 Evolution of the Pareto front throughout the epochs - 2^{nd} case study 60
6.7 Pareto front - 3^{rd} case study 61
6.8 Number of Pareto-optimal solutions and size of the VP at the end of each epoch 62
6.9 Pareto front - 4^{th} case study. The color of each point was defined using RGB code, where each parameter corresponds to the distance from the utopia point along each axis, normalized by the difference between the utopia and dystopia points. 63
6.10 Species representation on the Pareto front - 4^{th} case study 64
6.11 Position distribution per variable considering the Pareto-optimal solutions. Red markers indicate the mode of each variable's positions. 64

List of Symbols

θ	Shell laminates' ply angles
$\boldsymbol{\pi}$	Stacking sequences of the laminates
$\boldsymbol{\sigma}$	Stress vector
$\mathbf{h}$	Beam laminates' heights
$\mathbf{w}$	Beam laminates' widths
$\mathbf{A}$	Matrix of anisotropic parameters
$\mathbf{B}$	Strain matrix
$\mathbf{B}^e$	Element strain matrix
$\mathbf{D}$	Laminate equivalent stiffness matrix
$\mathbf{F}$	Vector of external forces
E	Strain energy
$\mathbf{T}$	Transformation matrix
$\mathbf{G}$	Combination of objective functions
$\mathbf{K}$	Stiffness matrix
$\mathbf{K}_\mathrm{S}$	Secant stiffness matrix
$\mathbf{K}_\mathrm{T}$	Tangent stiffness matrix
$\mathbf{K}^e$	Element stiffness matrix
$\overline{\mathbf{u}}$	Genotype
$\mathbf{u}$	Phenotype

List of Tables

4.1 Operator-*meme* correspondence 40

6.1 Mechanical properties of the composite materials used in the laminates 54
6.2 Material sequences 55
6.3 Parameter definitions for the MOMA 56
6.4 Summary of results - 1^{st} case study 57
6.5 Summary of results - 2^{nd} case study 58
6.6 Summary of results - 4^{th} case study 60

Listings

3.1 Fortran code section that checks whether ‘solution 2’ dominates ‘solution 1’ . . 29
3.2 Decoding function 30
3.3 Interface of Fortran function that defines the structure from problem variables . 30
4.1 Dynamic update of the VP 39
4.2 Stud generation. 47
5.1 Forward elimination using the `do concurrent` language construct 51

Chapter 1

Introduction

The advances in computer technology over the last decades have enabled the application of optimization techniques to solving increasingly complex problems in diverse engineering fields, which, in turn, has lead to greater efforts being allocated to the development of increasingly capable optimization algorithms (OAs) and frameworks that can leverage the available computational power.

In the context of engineering, optimization attempts to solve the problem of finding the best designs according to a set of performance metrics, while conforming to imposed constraints. Frequently, this process requires finding reasonable compromises between several contradicting objectives in a highly dimensional design space and hence constitutes a challenge which can only be systematically addressed through the use of OAs [3].

As far as structural engineering is concerned, the minimization of a system's material requirements often constitutes the most sensible approach to meeting sustainability targets. In this regard, fiber-reinforced plastic (FRP) composites are an attractive option given their durability, lower maintenance requirements and manageable environmental effects. Their specific strength and stiffness make them particularly attractive to the transportation sector. Given that fuel consumption can be decreased by reducing structural weight, irrespective of the fuel source, this is generally the most straightforward way of minimizing fuel consumption and therefore reducing CO_2 emissions [2].

Optimization methods can broadly be categorized into gradient-based methods and non-gradient-based methods, depending on the need to evaluate the derivatives of the objective and constraint functions in order to reach the optimal solution. From a domain approach standpoint, these methods can be classified as either point-based or population-based, where

the former rely on iterative techniques to define the search trajectory, while the latter employ evolutionary/generational strategies in which the fitness of the populations is continuously updated. Alternatively, optimization methods can be subdivided into classical and bio-inspired methods [3], the latter of which were made possible by the development of digital computers and resort to domain-specific knowledge to solve otherwise unsolvable problems in real time and with enough precision.

Evolutionary algorithms (EAs) are a class of general-purpose stochastic global optimization algorithms derived from Neo-Darwinism, a combination of the classical Darwinian evolution theory, the Weismann barrier and Mendelian genetics [4]. EAs may incorporate learning in order to accelerate evolution and they include Genetic Algorithms (GAs) [4] and Memetic Algorithms (MAs) [5].

1.1 Objectives

The primary objectives of this thesis can be summarized as follows:

- to formulate and develop an optimization model for composite structures based on MAs;
- to extend the MA so that it becomes a global optimization technique with convergence properties on global optima, by developing problem-based learning procedures and the associated genetic-cultural processes;
- to obtain empirical results to support the predictions offered by MA models using real scenario problems;
- to study the underlying finite element model and identify opportunities for improving the program.

1.2 Organization

Chapter 2 presents an overview of the evolution of Memetic Algorithms, addressing how they relate and are derived from Genetic Algorithms and the underlying concepts, namely Darwinian Evolution, the Selfish Gene Theory (or gene-centered view of evolution) and cultural evolution. The two classes of learning schemes present in MAs, Lamarckian learning and the Baldwin effect, as well as their effects on the global search, are then described, followed by an outline of the generations of MAs.

Chapter 3 describes the process of building the optimization model for hybrid composite

structures according to the Three Columns Concept, presenting the definition of the model of the system (the finite element model), the optimization model itself, including the objective functions, constraints and the multi-objective optimization problems (MOOPs) and concepts that it incorporates, and lastly the interface between the model of the system and the optimization algorithm, setting the context for the next chapter.

Chapter 4 focuses on the development of the MOMA, addressing its framework, operators and selection mechanisms.

Chapter 5 explains the improvements in efficiency made to the finite element program.

Chapter 6 discusses applications of the MOMA to four different MOOPs concerning the structural optimization of composite structures.

Chapter 7 presents the conclusions and future work perspectives.

Chapter 2

State of the Art

2.1 Genetic & Memetic algorithms: a synergistic view

Genetic algorithms (GAs) are a class of OAs inspired by the process of natural selection. They were formalized in Holland's 1975 book 'Adaptation in Natural and Artificial Systems' [6]. Holland's work showed how this evolutionary process can be applied to solve a wide variety of complex optimization problems using a highly parallel search technique. GAs have since been successfully applied in numerous domains, including machine learning, scientific modeling, and artificial life.

The No-Free-Lunch Theorem, introduced by Wolpert and Macready [7], states that the computational cost of finding a solution, averaged over all problems in the class, is the same for any solution method. In the field of computational intelligence, this perception has lead to the architectures of OAs being increasingly prioritized as opposed to the design of individual algorithms, with the aim of overcoming major drawbacks of evolutionary algorithms (EAs) and computational intelligence in general that stem from the fact that knowledge drawn from similar previous problems is not typically included in the OA [8].

Memetic algorithms (MAs) have been recently developed as a class of stochastic global search heuristics in which EA-based approaches are typically merged with heuristic local searches. The purpose of this hybridization is to find additional solutions that would otherwise be unreachable through either an evolutionary or local method alone [9]. Nonetheless, this conceptualization of MAs as mere search accelerators is rather simplistic, since the Selfish Gene Theory, presented by Richard Dawkins [10], introduced a novel interpretation of classical Darwinism where the genes themselves constitute the fundamental units of evolution, rather than individuals.

Through the lens of this gene-centered vision of evolution, natural selection occurs not at the level of organisms or populations, but rather at the level of the gene. Additionally, Dawkins introduced the term *meme* to designate a unit of cultural evolution, similar to how the gene is the unit of biological evolution. Such perspective indicates that cultural transmission may also be modelled by selfish replication, whereby the perceived utility or popularity of a given *meme* dictates its replication via inter-agent communication [10].

In 1989, Moscato introduced the MA [5], combining Darwin's principles of natural evolution and Dawkins' concept of *meme*. The dual nature of MAs reflects one of its defining features: the trade-off between the exploratory capabilities of GAs and the emphasis on exploitation typical of local-search strategies [11]. Accordingly, the global character of the search is mediated by the evolutionary nature of the approach, whereas intelligent local-search heuristics or other optimization methods mediate local search. Several iterations of the original MA have since been proposed, including the hybrid evolutionary algorithm, the genetic local search, the cultural algorithm, the Lamarckian evolutionary algorithm, the Baldwinian evolutionary algorithm and the diffusion memetic algorithm [8].

In practice, MAs (also known as population-based meta-heuristic search methods) make use of one or more populations that store both genotypes and *memes*, where the former represent solutions to a specific problem, while the latter indicate the strategies used to generate or improve those solutions. The synergy of this hybrid global-local approach is responsible for enhancing both exploration and exploitation during the search [12].

2.2 Lamarckian vs Baldwinian learning

In 1809's *Philosophie zoologique* [13], Lamarck proposed that the environment gives rise to changes in animals, which are subsequently inherited. This notion is reflected in Lamarckian learning, where any improvement obtained through local search is reflected in an individual's genotype, according to the inheritance mechanism of acquired traits. In contrast, the Baldwin effect [14] describes how evolution can be driven towards favourable adaptation without the referred inheritance and rather through changes in the fitness of individuals. Consequently, Baldwinian learning is associated with phenotypic plasticity, or an organism's ability to adapt to its environment [15, 16]. Both learning approaches have been used in recent investigations [9]. When a probabilistic combination of the two is adopted, the improved fitness of a given solution is always considered, whereas the genotype that corresponds to the improved fitness may replace the original one according to a given probability.

Lamarckian learning can accelerate the convergence of the evolutionary search, however, it may

lead to premature convergence of the OA [9], since diversity can be lost due to excessively pronounced local search. This situation can occur if, for example, every individual in a population assumes a local optimum using a complete local search, or in the case of a partial local search, if the design space under consideration has notably wide basins of attraction from which crossover and mutation cannot easily escape. Nonetheless, loss of genetic diversity can be offset when Baldwinian learning is introduced as well, though it may slow the convergence of the algorithm [9, 11, 17].

Learning provides faster adaptation to the environment compared to evolution, by enabling an individual to approach a local optimum in circumstances where its genotype did not grant such adaptation intrinsically, therefore increasing the individual's chance of survival. On the other hand, learning often spans an individual's lifetime and results in additional costs, which results in a pressure to select individuals whose genotypes grant them the desired adaptation from birth [18].

2.3 Generations of Memetic Algorithms

The first generation of MAs was marked by a hybridization between population-based global search, typical of EAs, and a cultural evolutionary stage. Despite their use of local optimizers, the first generation of MAs, introduced by Moscato in 1989 [5], did not apply the principles of Universal Darwinism to the transmission, variation and selection of memes. The typical design issues include [19]:

1. how often should individual learning be applied;
2. on which solutions should individual learning be used;
3. how long should individual learning be run;
4. what maximum computational budget to allocate for individual learning;
5. what individual learning method or *meme* should be used for a particular problem, sub-problem or individual.

The principles of memetic transmission and selection were first seen in the second generation of MAs [20], whose notable examples include the multi-meme [21], hyper-heuristic [22] and meta-Lamarckian [23, 24] MAs. The multi-meme MA encodes the memetic material, which defines the procedure to be used during local optimization, in the genotype of each individual. Dependent on an individual being selected, its memetic material is then inherited by the resulting offspring. In contrast, the hyper-heuristic and meta-Lamarckian MAs consider a pool

of candidate memes that compete for selection taking into account their performance history: a more successful *meme* will have greater chances of being replicated or copied.

Third generation MAs apply all three principles satisfying the definitions of a basic evolving system [20] and include co-evolution [25] and self-generation [26] MAs. In contrast to second generation MAs, which assumes the pool of memes to be used being known *a priori*, a rule-based representation of local search is co-adapted alongside candidate solutions within the evolutionary system, thus capturing regular repeated features or patterns in the problem space.

Meuth [8] proposes that a fourth generation of MAs should include the capability of the OA to not only learn during each separate problem instances, but accumulate knowledge between different optimization problems that belong to the same context. In this scenario, the meme selection process would benefit from the identification of recurring patterns, instead of starting with zero knowledge. This approach would constitute a brain inspired meta learning memetic computational system, possessing an optimizer, memory, a selection mechanism and a generalization mechanism that would conceptualize memes not just within the scope of a problem instance, but rather in a more generic contextual scope.

2.4 Recent developments in optimization of FRP structures

In an effort to reduce the costs associated with the manufacturing of FRP composite structures, considerable research has been devoted to the hybridization between expensive and higher stiffness materials and less expensive and lower stiffness materials. This approach is deemed adequate precisely due to it being a cost-effective solution to the problem of designing environmentally sustainable structures [2]. Pelletier and Vel [27] studied the minimization of weight of a graphite/epoxy pressure vessel combined with either loading capacity or stiffness maximization. A hierarchical genetic algorithm for multimodal optimization of composite structures was presented by Conceição António [28]. Rahul et al. [29] studied the minimization of weight and maximization of strength of hybrid laminates under impact loads. Baier et al. [30] optimize structures made of carbon composites and metal parts following strength and manufacturing requirements. A structural design and manufacturing model for composite materials was presented by Ghiasi et al. [31]. Irisarri et al. [32] proposed a multi-objective optimization formulation for rigid composite panels which resorts to design domain decomposition according to variables associated with the skin and reinforcements. Conceição António [33, 34] addressed sizing and topology in single and multi-objective optimization of hybrid composite structures.

Chapter 3

Building the optimization model for hybrid composite structures

The framework adopted in this project makes use of a hybrid construction technique that relies on the combination of materials that possess higher stiffness and are more expensive with ones that are less stiff and more affordable.

The objectives of the proposed optimal design approach are (Fig. 3.1):

- minimization of weight (Eq. 3.58),
- minimization of strain energy (Eq. 3.59),
- minimization of strain energy variability (Eq. 3.60).

These three objectives are addressed in two bi-objective optimization and one tri-objective optimization problems (2OOP and 3OOP, respectively):

- weight *vs* strain energy - which constitute two contradictory objectives [2] (Eq. 3.65),
- weight *vs* energy variability - also known as Robust Design Optimization (RDO). This methodology enables the systematic analysis and control of structural variability [2] (Eq. 3.66),
- weight *vs* strain energy *vs* energy variability - a case study introduced in the context of this thesis (Eq. 3.67).

The present model performs the trade-off between performance targets under stress, displacement, and buckling constraints.

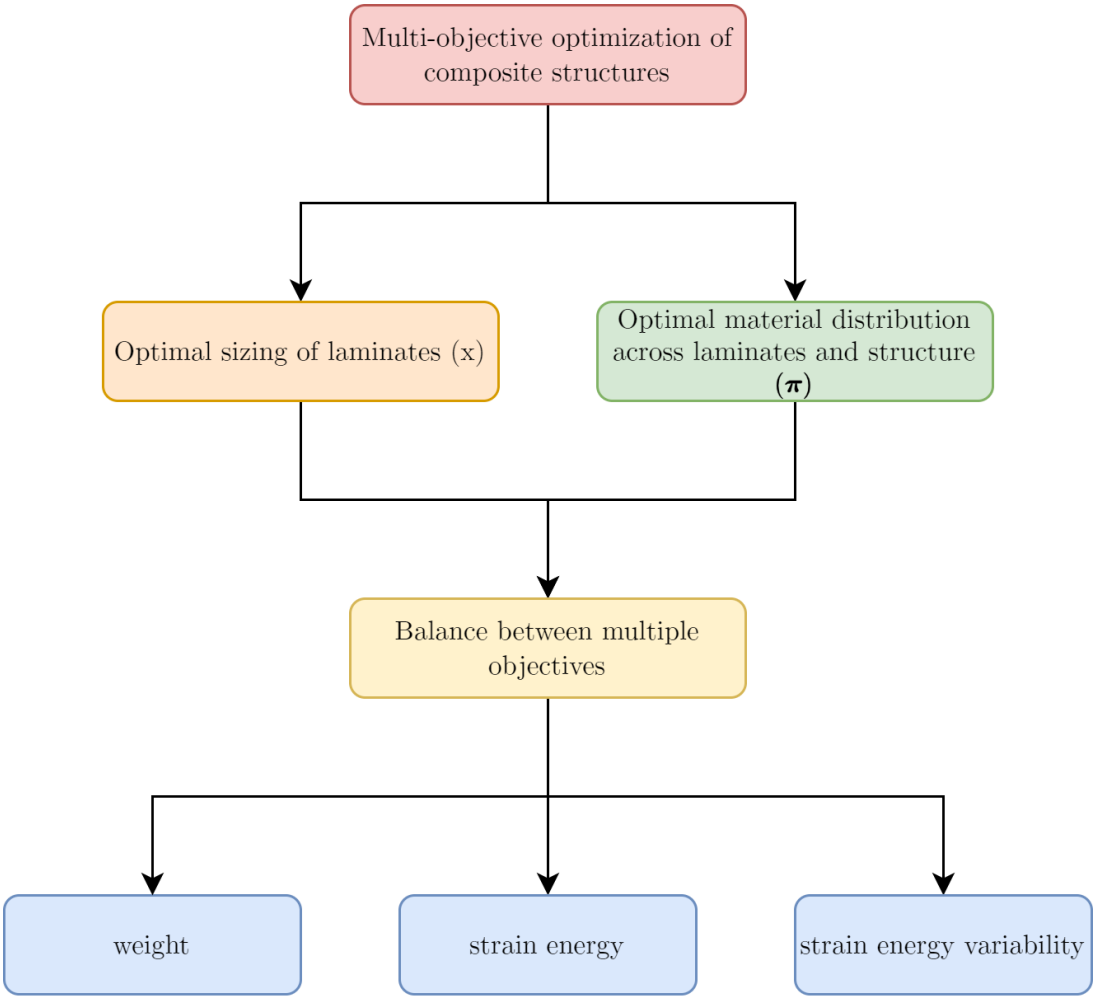


Figure 3.1: Framework for the design of hybrid composite structures

3.1 Three Columns approach to a MOOP

An optimization study is typically carried out following the development and validation of a simulation model. In order to transform an analytical resource into a design resource, optimization concepts are utilized. Such a study should typically begin with the simplest possible model, which can be achieved via meta-models. These are derived from more complex analysis models through a range of data manipulation techniques and enable the simplification of the problem. Afterwards, these meta-models can then be applied to the first optimization experiments [3].

The optimization problem is handled according to the Three Columns Concept integrated into the design optimization process (Fig. 3.2), which establishes three entities: the system analysis model, the optimization algorithm and the optimization model [3].

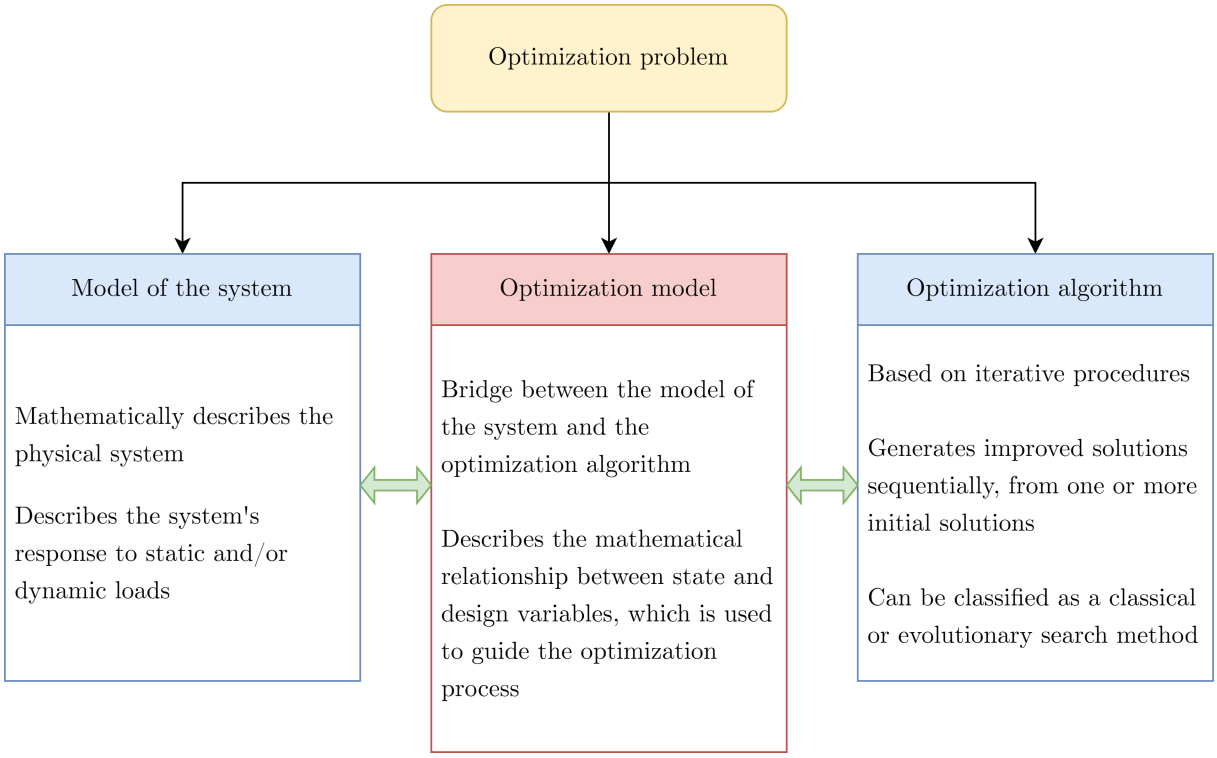


Figure 3.2: Three columns approach to an optimization problem

First Column: System analysis model

A cornerstone of an optimization process is the creation of a representative model for a particular physical system, which constitutes the mathematical description of its behaviour. In the context of mechanical systems, such a model generally describes the system's response to static and/or dynamic loads in terms of strains, stresses, eigenfrequencies, buckling loads, displacements, velocities, etc. and is used to evaluate each design solution. Since there is a direct or indirect relationship between the design variables and the configuration of the system parameters, their definition is conditioned: changes in the system modify the values of the state variables (stresses, strains, displacements, etc.) that are updated for the current solution.

Second Column: Optimization algorithm

Optimization algorithms are based on iterative procedures that generate a sequence of improved design solutions either from a single initial solution or a population of solutions. When a predetermined convergence criterion is met, the process is terminated. In structural optimization, a third class of global optimum search methods, known as evolutionary search methods, has recently surfaced. These do not employ gradients and are instead based on Darwin's Theory.

Third Column: Optimization model

The optimization model constitutes a bridge between the analysis model and the optimization algorithm, by establishing the mathematical relationship between state and design variables. The former are dependent on the system's model parameters, which change during the optimization process. Ultimately, the optimization model performs parameterization through variable definitions and transformations, one such example being GAs, which mostly operate on genotype variables that can be encoded as strings of *bits*, which are then transformed into a phenotype of design variables.

3.2 Model of the system: Finite element model

The present structural analysis of hybrid composites resorts to a FEM formulation previously proposed by Conceição António [28] and considers beam-reinforced shallow shell structures that exhibit nonlinear geometric behavior. The structure is discretized using Marguerre shell elements and Timoshenko beam elements (Fig. 3.3).

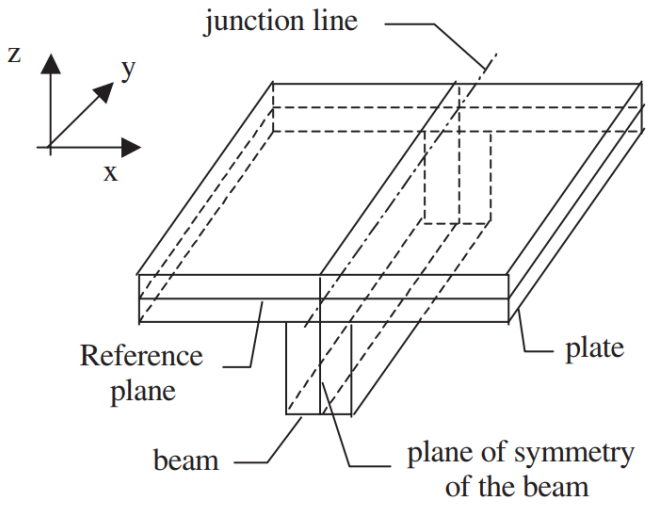


Figure 3.3: Beam-reinforced shallow shell structure [1]

3.2.1 Finite element formulation

The displacement field vector of the Marguerre shell element, which originates from the combination of the Marguerre and Mindlin plate theories [35], is defined as

$$\mathbf{d} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \bar{u}(x,y) - z\theta_x(x,y) \\ \bar{v}(x,y) - z\theta_y(x,y) \\ \bar{w}_1(x,y) + \bar{w}_0(x,y) \end{bmatrix} \tag{3.1}$$

where

- $\bar{u}, \bar{v}, \bar{w}_1$ are the reference surface translations,
- θ_x and θ_y are the rotations of the normal to the planes $x - z$ and $y - z$, respectively,
- $\bar{w}_0$ is the deflection of the plate in its initial (undeformed) configuration.

The strain vector is obtained by applying the Mindlin and von Kármán hypotheses to the Green deformation tensor and is defined in the global coordinate system as [35]

$$\varepsilon = \begin{bmatrix} \varepsilon_x & \varepsilon_y & \gamma_{xy} & \gamma_{xz} & \gamma_{yz} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \varepsilon_p \\ \varepsilon_s \end{bmatrix} = \begin{bmatrix} \varepsilon_m + z\varepsilon_b + \varepsilon_i + \varepsilon_{NL} \\ \varepsilon_s \end{bmatrix} \tag{3.2}$$

where

- $\varepsilon_m, \varepsilon_b, \varepsilon_s$ are the membrane, bending and shear strains, respectively,
- ε_{NL} contains the nonlinear components of the in-plane strains ε_p,
- ε_i is the linear contribution of the initial deformation to the in-plane strains.

The constitutive equations are derived for an orthotropic material with material axes $(1, 2)$ parallel to the xy plane and at an angle θ with respect to the x and y, with the material axis 3 being parallel to z. For a given layer of the element, the stresses relate to the deformations as [35]

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \sigma_y & \tau_{xy} & \tau_{xz} & \tau_{yz} \end{bmatrix}^{\mathrm{T}} = \mathbf{D}\boldsymbol{\varepsilon} = \mathbf{T}^{\mathrm{T}}\bar{\mathbf{D}}\mathbf{T}\boldsymbol{\varepsilon} \tag{3.3}$$

where

- $\mathbf{T}$ is the strain transformation matrix,
- $\bar{\mathbf{D}}$ is the material elasticity matrix with respect to the material axes [36].

It is assumed that the beam element is symmetric about the local $x'z$ plane and weak in bending relative to the displacements parallel to the plane of the plate, so that the corresponding bending

stiffness and local buckling of the stiffeners and cross-sectional strains can be neglected. A local axes system (x', y', z), where the $x'y'$ plane is parallel to the global xy plane, is used to model the beam element's structural behavior.

The displacements of a point (x', y', z) that belongs to the stiffener are

$$\bar{u} = \bar{\hat{u}}(x') - z\theta_{x'}(x'), \quad w = \hat{w}_1(x') + \hat{w}_0(x') \tag{3.4}$$

where

- $\bar{\hat{u}}$ is the axial displacement of the centroid,
- $\theta_{x'}$ is the rotation about y' local axis,
- the deflection $\hat{w}_0$ corresponds to the undeformed configuration of the beam and $\hat{w}_1$ is due to its deformation.

Assuming that the local axes x', y' of the beam do not coincide with the reference axes x, y and are instead rotated at an angle α, the displacements of the centroid of the beam, $\bar{\mathbf{d}}_i = [\bar{\hat{u}}, \hat{w}_1, \theta_{x'}, \theta_{y'}]_i^{\mathrm{T}}$, are expressed in terms of the global displacements of the adjacent plate's i^{th} node, $\mathbf{d}_i = [\hat{u}, \hat{v}, \hat{w}_1, \theta_x, \theta_y]_i^{\mathrm{T}}$, as

$$\bar{\hat{u}}_i = \hat{u}_i \cos\alpha + \hat{v}_i \sin\alpha \tag{3.5}$$

$$\begin{bmatrix} \theta_{x'} \\ \theta_{y'} \end{bmatrix}_i = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_y \end{bmatrix}_i \tag{3.6}$$

The strain vector of the beam element is defined as

$$\boldsymbol{\varepsilon}' = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \varepsilon_t \end{bmatrix} = \begin{bmatrix} \varepsilon_{x'} \\ \gamma_{x'z} \\ \varepsilon_t \end{bmatrix} = \begin{bmatrix} \varepsilon_m + z\varepsilon_b + \varepsilon_i \\ \gamma_{x'z} \\ \varepsilon_t \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \bar{\hat{u}}}{\partial x'} - z\dfrac{\partial \theta_{x'}}{\partial x'} + \dfrac{\partial \hat{w}_1}{\partial x'}\dfrac{\partial \hat{w}_0}{\partial x'} \\ \dfrac{\partial \hat{w}_1}{\partial x'} - \theta_{x'} \\ \dfrac{\partial \theta_{y'}}{\partial x'} \end{bmatrix} \tag{3.7}$$

in which ε_t is the torsional strain due to the variation of the axial rotation $\theta_{y'}$ along the x' axis.

In order to derive the stress-strain relations for the beam element, it is assumed that its local

axes coincide with the material axes, $1 \equiv x'$, $2 \equiv y'$ and $3 \equiv z$. Assuming then that $\sigma_{x'}$ and $\tau_{x'z}$ are the only non-zero terms of the stress vector $\boldsymbol{\sigma}$,

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{x'} \\ \tau_{x'z} \end{bmatrix} = \mathbf{D} \begin{bmatrix} \varepsilon_{x'} \\ \gamma_{x'z} \end{bmatrix} = \mathbf{D}\boldsymbol{\varepsilon} \tag{3.8}$$

whereas the term ε_t induces a torsional moment $M_{y'}$ which, for a homogeneous material, is given by

$$M_{y'} = GJ\varepsilon_t \tag{3.9}$$

where GJ is the torsional rigidity of the stiffener.

3.2.2 Finite element discretization

The discretization process resorts to a nine node shell and a three node beam elements. The beam elements are assumed to be monolithically connected to the borders of the adjacent shell elements, which implies that identical displacement fields are produced at these connections. Both shell and beam elements are divided into layers throughout the thickness, so that each layer contains Gauss integration points in its midsection. The stress components and stiffness contributions are evaluated at these points and assumed to be constant along each layer's thickness. The stress distribution is modelled by a piece-wise constant approximation of the integral over the thickness using a mid-ordinate rule, where the variation of a given material property is approximated by a number of discrete values corresponding to the number of layers [37].

The displacement $\hat{\mathrm{u}}$ within an element is a function of n discrete nodal displacements:

$$\hat{\mathbf{u}} = \begin{bmatrix} \hat{\mathrm{u}} \\ \hat{\mathrm{v}} \\ \hat{\mathrm{w}}_1 \\ \theta_x \\ \theta_y \end{bmatrix} = \sum_{i}^{n} [N_i \, \mathbf{I}_s] \, \delta_i = \sum_{i=1}^{n} \begin{bmatrix} \mathrm{N_i} & 0 & 0 & 0 & 0 \\ 0 & \mathrm{N_i} & 0 & 0 & 0 \\ 0 & 0 & \mathrm{N_i} & 0 & 0 \\ 0 & 0 & 0 & \mathrm{N_i} & 0 \\ 0 & 0 & 0 & 0 & \mathrm{N_i} \end{bmatrix} \begin{bmatrix} \hat{\mathrm{u}}_i \\ \hat{\mathrm{v}}_i \\ \hat{\mathrm{w}}_i \\ \theta_{xi} \\ \theta_{yi} \end{bmatrix} \tag{3.10}$$

where N_i are the shape functions.

The current project adopts a Lagrangian formulation for the shell elements. For an isoparametric

element, the shape functions are [35]

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\xi)(1-\eta)(2-\xi-\eta), \quad N_5 = \frac{1}{2}(1-\xi^2)(1-\eta) \\ N_2 &= \frac{1}{4}(1+\xi)(1-\eta)(2+\xi-\eta), \quad N_6 = \frac{1}{2}(1+\xi)(1-\eta^2) \\ N_3 &= \frac{1}{4}(1+\xi)(1+\eta)(2+\xi+\eta), \quad N_7 = \frac{1}{2}(1-\xi^2)(1+\eta) \\ N_4 &= \frac{1}{4}(1-\xi)(1+\eta)(2-\xi+\eta), \quad N_8 = \frac{1}{2}(1-\xi)(1-\eta^2) \\ & N_9 = (1-\xi^2)(1-\eta^2) \end{aligned} \tag{3.11}$$

where ξ and η are the natural coordinates within the element. Eq. 3.10 can be rewritten as [35]

$$\hat{\mathbf{u}} = \mathbf{N}\boldsymbol{\delta}, \quad \boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_1 & \boldsymbol{\delta}_2 & \dots & \boldsymbol{\delta}_n \end{bmatrix}^{\mathrm{T}} \tag{3.12}$$

The virtual displacements are written in terms of the nodal virtual displacements as [35]

$$\mathrm{d}\hat{\mathbf{u}} = \mathbf{N}\mathrm{d}\boldsymbol{\delta}, \quad \mathrm{d}\boldsymbol{\delta} = \mathrm{d}\begin{bmatrix} \boldsymbol{\delta}_1 & \boldsymbol{\delta}_2 & \dots & \boldsymbol{\delta}_n \end{bmatrix}_i^{\mathrm{T}} \tag{3.13}$$

The displacement gradients $\boldsymbol{\theta}$ can then be written in terms of the nodal displacements $\boldsymbol{\delta}$ and cartesian derivatives of the shape functions as [35]

$$\boldsymbol{\theta} = \mathbf{G}\boldsymbol{\delta}, \quad \mathbf{G} = \begin{bmatrix} G_1 & G_2 & \dots & G_n \end{bmatrix}, \quad \mathbf{G}_i = \begin{bmatrix} 0 & 0 & \dfrac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & 0 & \dfrac{\partial N_i}{\partial y} & 0 & 0 \end{bmatrix} \tag{3.14}$$

The virtual gradients are now written in terms of virtual nodal displacements as [35]

$$\mathrm{d}\boldsymbol{\theta} = \mathbf{G}\mathrm{d}\boldsymbol{\delta} \tag{3.15}$$

The generalised Green strain vector $\hat{\varepsilon}$ depends on the nodal displacements δ, displacement gradients $\mathbf{A}_\theta$ and Cartesian derivatives of $\mathbf{N}$ as

$$\hat{\varepsilon} = \left[\mathbf{B}_0 + \frac{1}{2}\mathbf{B}_L(\boldsymbol{\delta})\right]\boldsymbol{\delta} \tag{3.16}$$

where

- $\mathbf{B}_0$ is the strain matrix related to the linear strains ε_p^0, ε_b^0 and ε_t^0,
- $\mathbf{B}_L$, which is linearly dependent on δ, gives the nonlinear strains ε_p^L. Consequently the nonlinear strains are quadratically dependent on the nodal displacements, $\boldsymbol{\delta}$.

The constant matrix $\mathbf{B}_0$ is written in terms of nodal submatrices as

$$\mathbf{B}_0 = \begin{bmatrix} \mathbf{B}_{01} & \mathbf{B}_{02} & \dots & \mathbf{B}_{0n} \end{bmatrix} \tag{3.17}$$

where

$$\mathbf{B}_{0i} = \begin{bmatrix} \mathbf{B}_0^p & 0 \\ 0 & \mathbf{B}_{0i}^b \\ 0 & \mathbf{B}_{0i}^s \end{bmatrix}$$

$$\mathbf{B}_0^p = \begin{bmatrix} \dfrac{\partial N_i}{\partial x} & 0 \\ 0 & \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial y} & \dfrac{\partial N_i}{\partial x} \end{bmatrix}, \quad \mathbf{B}_{0i}^b = \begin{bmatrix} 0 & -\dfrac{\partial N_i}{\partial x} & 0 \\ 0 & 0 & -\dfrac{\partial N_i}{\partial y} \\ 0 & -\dfrac{\partial N_i}{\partial y} & -\dfrac{\partial N_i}{\partial x} \end{bmatrix}, \quad \mathbf{B}_{0i}^s = \begin{bmatrix} \dfrac{\partial N_i}{\partial x} & -N_i & 0 \\ \dfrac{\partial N_i}{\partial y} & 0 & -N_i \end{bmatrix} \tag{3.18}$$

The nonlinear component of the strain matrix is built from the corresponding nodal submatrices

$$\mathbf{B}_L = \begin{bmatrix} \mathbf{B}_L^1 & \mathbf{B}_L^2 & \dots & \mathbf{B}_L^n \end{bmatrix}, \quad \mathbf{B}_L^i = \begin{bmatrix} \mathbf{A}_{\boldsymbol{\theta}} \\ 0 \\ 0 \end{bmatrix} \mathbf{G}_i \tag{3.19}$$

3.2.3 Solution method

The equilibrium equations are derived from a displacement finite element discretization based on the Total Lagrangian formulation, where the nodal displacements, $\mathbf{d}$, are referred to the initial configuration of the structure, and the loading is assumed to be conservative. For a load level t, this set of nonlinear equilibrium equations can be written as [37]

$$\Psi(\mathbf{d}, \lambda) = \int_V \mathbf{B}^{\mathrm{T}} \sigma \, \mathrm{d}V + \int_L \mathbf{B}_{\mathrm{t}}{}^{\mathrm{T}} M_{y'} \, \mathrm{d}L - \lambda \mathbf{F} = \mathbf{r}(\mathbf{d}) - \lambda \mathbf{F} = \mathbf{0}, \tag{3.20}$$

where

- $\mathbf{r}(\mathbf{d})$ are the equivalent nodal forces due to the stress distribution,
- $\mathbf{F}$ is the equivalent nodal force vector due to external applied loads,
- $\mathbf{B}$ is the strain matrix, dependent on the nodal displacements, $\mathbf{d}$,
- the line integral denotes the effect of torsional strain in the beam section on the total deformation energy of the structure.

The Newton-Raphson method is used to obtain the solution to Eq. 3.20 [37], by iteratively approximating the equations through a series of linear solutions so as to reduce the errors transferred to the next load level. Denoting by $\mathbf{d}_i$ an estimate of the nodal displacements for the i^{th} iteration of a load step λ_i, the corresponding residual forces are $\Psi(\mathbf{d}_i, \lambda_i) \neq \mathbf{0}$ and an improved solution $\mathbf{d}_{i+1}$ for the displacements is obtained as

$$\mathbf{d}_{i+1} = \mathbf{d}_i + \delta\mathbf{d}_i = \mathbf{d}_0 + \Delta\mathbf{d}_i \tag{3.21}$$

where $\mathbf{d}_0$ is the displacement solution at the end of the previous increment,

$$\delta\mathbf{d}_i = -\left.\frac{\partial\Psi(\mathbf{d})}{\partial\mathbf{d}}\right|_i \Psi(\mathbf{d}_i) = -\mathbf{K}_{\mathrm{T}i}^{-1}\Psi(\mathbf{d}_i) \tag{3.22}$$

is the iterative variation of the displacements, $\mathbf{K}_{\mathrm{T}i}$ is the assembled tangent stiffness matrix of the structure and $\Delta\mathbf{d}_i$ are the incremental displacements.

Arc-length method

During the optimization process and under particular geometric instability conditions, where 'snap-through' and 'snap-back' phenomena may occur, the Newton-Raphson method is incapable of traversing the equilibrium path beyond the limit point. This motivated the development of continuation methods based on incremental predictor-corrector solution procedures. The spherical formulation of the arc-length method, presented by Crisfield [38], is implemented, which forces the iterative process to remain on a 'sphere' of radius Δl, centered on the converged solution of the previous load level [37]. The variation of nodal displacements $\delta\mathbf{d}_i$ can be written as

$$\delta\mathbf{d}_i = -\mathbf{K}_i^{-1}\,\Psi(\lambda_i) + \delta\lambda_i\,\mathbf{K}_i^{-1}\,\mathbf{F} = \overline{\delta\mathbf{d}}_i + \delta\lambda_i{}^t\mathbf{d}_i \tag{3.23}$$

where

- λ_i is the load level factor for the i^{th} iteration,
- $\overline{\delta \mathbf{d}}_i$ the variation of the nodal displacement related to $\Psi(\lambda_i)$, as obtained in Eq. 3.22,
- ${}^t\mathbf{d}_i$ are the nodal displacements due to the total reference load $\mathbf{F}$.

In order to obtain the iterative variation of the applied load factor, $\delta\lambda_i$, the constrained equation of the 'spherical' arc-length method

$$\Delta\mathbf{d}_{i+1}{}^{\mathrm{T}}\Delta\mathbf{d}_{i+1} + b\Delta\lambda_{i+1}{}^2\mathbf{F}^{\mathrm{T}}\mathbf{F} = \Delta l^2 \tag{3.24}$$

is considered as suggested by Crisfield [38] with $b = 0$. Using the updated relations

$$\Delta\lambda_{i+1} = \Delta\lambda_i + \delta\lambda_i \tag{3.25}$$

$$\Delta\mathbf{d}_{i+1} = \Delta\mathbf{d}_i + \delta\mathbf{d}_i = \overline{\Delta\mathbf{d}}_i + \delta\lambda_i{}^t\mathbf{d}_i \tag{3.26}$$

the following quadratic equation is found:

$$u(\mathbf{d}, \lambda) = {}^t\mathbf{d}_i{}^{\mathrm{T}}\,{}^t\mathbf{d}_i(\delta\lambda_i)^2 + 2\overline{\Delta\mathbf{d}}_i{}^{\mathrm{T}}\,{}^t\mathbf{d}_i\,\delta\lambda_i + \overline{\Delta\mathbf{d}}_i{}^{\mathrm{T}}\,\overline{\Delta\mathbf{d}}_i - \Delta l^2 = 0 \tag{3.27}$$

with

$$\overline{\Delta\mathbf{d}}_i = \Delta\mathbf{d}_i + \overline{\delta\mathbf{d}}_i \tag{3.28}$$

The iterative variation of the applied load factor $\delta\lambda_i$ is obtained by solving Eq. 3.27. To avoid 'doubling back' on the equilibrium path, the solution for $\delta\lambda_i$ that corresponds to the maximum positive value of Eq. 3.29 must be chosen.

$$\chi = \Delta\mathbf{d}_{i+1}{}^{\mathrm{T}}\,\Delta\mathbf{d}_i \tag{3.29}$$

For all load increments other than the first, the initial incremental loading parameter $\Delta\lambda_i$ are given by

$$\Delta\lambda_1 = \pm\Delta l \sqrt{{}^t\mathbf{d}_1{}^{\mathrm{T}}\,{}^t\mathbf{d}_1} \tag{3.30}$$

The sign of Eq. 3.30 must be the same as the one from the previous increment, unless the determinant of $\mathbf{K}$ changes sign, in which case a sign reversal is performed.

This procedure can be described as augmenting Eq. 3.20 with the constraint function $u(\mathbf{d}, \lambda)$ (Eq. 3.27):

$$\begin{bmatrix} \Psi(\mathbf{d}, \lambda) \\ u(\mathbf{d}, \lambda) \end{bmatrix} = \mathbf{0} \tag{3.31}$$

3.2.4 Structural integrity analysis

The data obtained from solving the equilibrium equation (Eq. 3.31) is used to define the limit state functions, which depend on the vector of design variables, $\mathbf{u} \in \mathbb{R}^n$, in order to evaluate the structural integrity. This procedure unifies buckling and first ply failure (FPF) analysis based on the concepts of critical load factor and critical displacement [1].

Critical load factor

Once the equilibrium path is established, the critical points can be determined through the approach proposed by Budiansky [39], where the equilibrium at a critical point $(\mathbf{d}_b, \lambda_b)$ is defined as

$$\Psi(\mathbf{d}_b, \lambda_b)\ \delta\mathbf{d} = \mathbf{0} \tag{3.32}$$

whose first order approximation is

$$\nabla_{\mathbf{d}}\ \Psi(\mathbf{d}_b, \lambda_b)\ \Delta\mathbf{d}\ \delta\mathbf{d} = \mathbf{K}_{\mathrm{T}}(\mathbf{d}_b, \lambda_b)\ \phi = \mathbf{0} \tag{3.33}$$

where ϕ and $\mathbf{K}_{\mathrm{T}}$ represent the buckling mode and tangent stiffness matrix, respectively. Alternatively, since condition 3.33 is equivalent to the singularity of the tangent stiffness matrix, it can be replaced by

$$\det(\mathbf{K}_{\mathrm{T}}(\mathbf{d}_b, \lambda_b)) = 0 \tag{3.34}$$

Considering the linearized buckling analysis as 3.34 and supposing that the critical point $(\mathbf{d}_b, \lambda_b)$ is localized at load level t, it can be assumed that

$$\mathbf{K}_{\mathrm{T}}(\mathbf{d}_b, \lambda_b) = \mathbf{K}_{\mathrm{T}}({}^{t-1}\mathbf{d}, {}^{t-1}\lambda) + \xi \left[\mathbf{K}_{\mathrm{T}}({}^{t}\mathbf{d}, {}^{t}\lambda) - \mathbf{K}_{\mathrm{T}}({}^{t-1}\mathbf{d}, {}^{t-1}\lambda)\right], \tag{3.35}$$

where ξ is a scalar quantity. The buckling load λ_b can then be approximated as

$$\lambda_b = {}^{t-1}\lambda + \xi({}^{t}\lambda - {}^{t-1}\lambda) \tag{3.36}$$

Due to the fact that Eqs. 3.35 and 3.36 assume proportionality between the stiffness and the load increment, the buckling load may be overestimated. Nevertheless, the previously defined eigenvalue problem can be used to calculate the critical load and the maximum load vector in Eq. 3.36 given from the respective lowest eigenvalue and its related eigenvector.

The concept of FPF based on Tsai's parameter [40], commonly used in linear behavior, cannot be directly applied to the nonlinear analysis of composite structures. Alternatively, an approach that relies on the FPF load factor, λ_{FPF}, is considered. This load factor is obtained from Eq. 3.35, using a similar scheme to the one proposed in Eq. 3.36. Supposing that the critical point $(\mathbf{d}_{\mathrm{FPF}}, \lambda_{\mathrm{FPF}})$ occurs for load level t, the approximation

$$\lambda_{\mathrm{FPF}} = {}^{t-1}\lambda + \gamma({}^{t}\lambda - {}^{t-1}\lambda), \tag{3.37}$$

where γ is an appropriate scalar, is used to identify the first ply failure load factor, λ_{FPF}, and the corresponding critical displacement, $\mathbf{d}_{\mathrm{FPF}}$, on the equilibrium path. The failure criterion for anisotropic materials is a generalized form of the Huber-Mises law and can be written as

$$\frac{f_k(\boldsymbol{\sigma})}{Y} - 1 = 0, \; k = 1, \ldots, N_s, \tag{3.38}$$

where Y is the maximum allowable stress level, N_s is the number of points where the stress vector is evaluated and $f_k(\boldsymbol{\sigma})$ is a failure function evaluated at the k^{th} point, defined in terms of the stresses relative to the material axes $1, 2, 3$ as

$$f_k^2(\boldsymbol{\sigma}) = a_1\sigma_1^2 + 2a_{12}\sigma_1\sigma_2 + a_2\sigma_2^2 + a_3\tau_{12}^2 + a_4\tau_{13}^2 + a_5\tau_{23}^2 = \boldsymbol{\sigma}_{1,2,3}{}^{\mathrm{T}}\, \mathbf{A}\, \boldsymbol{\sigma}_{1,2,3} \tag{3.39}$$

where $\mathbf{A}$ is the matrix of anisotropic strength parameters which are obtained from six

independent failure tests. If the material axes 1,2 are rotated at an angle θ relative to the reference axes x, y, transforming both the stresses and matrix $\mathbf{A}$ to the global axes is required. The obtained yield criterion is then rewritten as

$$f_k^2(\boldsymbol{\sigma}) = \boldsymbol{\sigma}^{\mathrm{T}}\, \mathbf{T}^{\mathrm{T}}\, \mathbf{A}\, \mathbf{T}\, \boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathrm{T}}\, \bar{\mathbf{A}}\, \boldsymbol{\sigma} \tag{3.40}$$

where $\mathbf{T}$ is the strain transformation matrix that relates the material system $(1, 2, 3)$ to the global system (x, y, z). Finally, FPF occurs for

$$h_{\mathrm{FPF}} = \max\left(\frac{f_k(\boldsymbol{\sigma})}{Y} - 1,\ k = 1, \ldots, N_s\right) = 0 \tag{3.41}$$

The previous buckling and FPF-based analysis is used to establish the unified approach considering the critical load factor, λ_{crit}, defined as

$$\lambda_{crit}(\mathbf{u}) = \min\left(\lambda_b(\mathbf{u}),\ \lambda_{\mathrm{FPF}}(\mathbf{u})\right) \tag{3.42}$$

and the associated limit state function, where $\bar{\lambda}_a$ is the allowable load factor prescribed for critical load factor

$$\varphi_1(\mathbf{u}) = 1 - \frac{\lambda_{crit}(\mathbf{u})}{\bar{\lambda}_a} \tag{3.43}$$

Critical displacement

In this approach, the most critical displacement, d_{crit}, is either the critical displacement, $\mathbf{d}_b$, associated with the first limit point, or the critical displacement, $\mathbf{d}_{\mathrm{FPF}}$, associated with the FPF load factor, and is defined as

$$d_{crit}(\mathbf{u}) = \max\left(\mathbf{d}_b(\mathbf{u}),\ \mathbf{d}_{\mathrm{FPF}}(\mathbf{u})\right) \tag{3.44}$$

The corresponding limit state function concerns to the critical nodal displacement d_{crit} exceeding an allowable $\bar{d}_a$ as

$$\varphi_2(\mathbf{u}) = \frac{d_{crit}(\mathbf{u})}{\bar{d}_a} - 1 \tag{3.45}$$

3.2.5 Approximation of the strain energy gradient

The current framework addresses the minimization of the strain energy variability, hence the definition of the energy gradient of the structure becomes imperative. The energy gradient can be evaluated using the direct differentiation method [41]. Considering the nonlinear equilibrium equation of the structural system for a load factor ${}^{n+1}\lambda$,

$$\mathbf{K}_{\mathrm{S}}{}^{n+1}\mathbf{d} = {}^{n+1}\lambda\, \mathbf{f}_{ext} \tag{3.46}$$

where $\mathbf{K}_{\mathrm{S}}$ is the secant stiffness matrix and ${}^{n+1}\mathbf{d}$ is the displacement vector, the strain energy can be approximated as

$$\tilde{U} = \frac{1}{2}\, {}^{n+1}\mathbf{d}^{\mathrm{T}}\, \mathbf{K}_{\mathrm{S}}\, {}^{n+1}\mathbf{d} \tag{3.47}$$

and the derivative of elastic energy with respect to the design variable x_i is written as

$$\frac{\partial \tilde{U}}{\partial x_i} = \frac{1}{2}\left[\left(\frac{\partial^{n+1}\mathbf{d}}{\partial x_i}\right)^{\mathrm{T}} \mathbf{K}_{\mathrm{S}}\, {}^{n+1}\mathbf{d} + {}^{n+1}\mathbf{d}^{\mathrm{T}}\, \frac{\partial \mathbf{K}_{\mathrm{S}}}{\partial x_i}\, {}^{n+1}\mathbf{d} + {}^{n+1}\mathbf{d}^{\mathrm{T}}\, \mathbf{K}_{\mathrm{S}}\, \frac{\partial^{n+1}\mathbf{d}}{\partial x_i}\right] \tag{3.48}$$

Since the secant stiffness matrix has symmetry properties,

$$\frac{\partial^{n+1}\mathbf{d}}{\partial x_i}^{\mathrm{T}} (\mathbf{K}_{\mathrm{S}}\, {}^{n+1}\mathbf{d}) = (\mathbf{K}_{\mathrm{S}}\, {}^{n+1}\mathbf{d})^{\mathrm{T}}\, \frac{\partial^{n+1}\mathbf{d}}{\partial x_i} \tag{3.49}$$

$$= {}^{n+1}\mathbf{d}^{\mathrm{T}}\, \mathbf{K}_{\mathrm{S}}{}^{\mathrm{T}}\, \frac{\partial^{n+1}\mathbf{d}}{\partial x_i} \tag{3.50}$$

$$= {}^{n+1}\mathbf{d}^{\mathrm{T}}\, \mathbf{K}_{\mathrm{S}}\, \frac{\partial^{n+1}\mathbf{d}}{\partial x_i} \tag{3.51}$$

and Eq. 3.48 can then be rewritten as

$$\frac{\partial \tilde{U}}{\partial x_i} = \frac{1}{2}\, {}^{n+1}\mathbf{d}^{\mathrm{T}}\, \frac{\partial \mathbf{K}_{\mathrm{S}}}{\partial x_i}\, {}^{n+1}\mathbf{d} + {}^{n+1}\mathbf{d}^{\mathrm{T}}\, \mathbf{K}_{\mathrm{S}}\, \frac{\partial^{n+1}\mathbf{d}}{\partial x_i} \tag{3.52}$$

Eq. 3.52 can be simplified by differentiating Eq. 3.46:

$$\frac{\partial \mathbf{K}_{\mathrm{S}}}{\partial x_i}\,{}^{n+1}\mathbf{d} + \mathbf{K}_{\mathrm{S}}\frac{\partial^{n+1}\mathbf{d}}{\partial x_i} = \frac{\partial^{n+1}\lambda \mathbf{f}_{ext}}{\partial x_i} = 0 \tag{3.53}$$

Assuming that the nodal load vector does not depend on the design variables,

$$\mathbf{K}_{\mathrm{S}}\frac{\partial^{n+1}\mathbf{d}}{\partial x_i} = -\frac{\partial \mathbf{K}_{\mathrm{S}}}{\partial x_i}\,{}^{n+1}\mathbf{d} \tag{3.54}$$

and the derivative of the elastic strain energy can be simplified as

$$\frac{\partial \tilde{U}}{\partial x_i} = -\frac{1}{2}\,{}^{n+1}\mathbf{d}^{\mathrm{T}}\,\frac{\partial \mathbf{K}_{\mathrm{S}}}{\partial x_i}\,{}^{n+1}\mathbf{d} \tag{3.55}$$

Since $\mathbf{K}_T$ is already known from the incremental and iterative process associated with the solution algorithm (3.2.3), its derivative can be used to approximate the derivative of $\mathbf{K}_S$:

$$\frac{\partial \mathbf{K}_{\mathrm{S}}}{\partial x_i}\,{}^{n+1}\mathbf{d} \approx \sum_{k=1}^{n+1}\left(\frac{\partial \mathbf{K}_{\mathrm{T}}}{\partial x_i}\right)_k \Delta \mathbf{d}_i \tag{3.56}$$

Substituting Eq. 3.56 into Eq. 3.55 finally gives

$$\frac{\partial \tilde{U}}{\partial x_i} \approx -\frac{1}{2}\,{}^{n+1}\mathbf{d}^{\mathrm{T}}\left[\sum_{k=1}^{n+1}\left(\frac{\partial \mathbf{K}_{\mathrm{T}}}{\partial x_i}\right)_k \Delta \mathbf{d}_k\right] \tag{3.57}$$

3.3 Optimization model for the MOOP

A laminated beam-reinforced shell structure is adopted in the current setting (Fig. 3.4), whose sizing design variables, grouped in vector $\mathbf{x}$, consist of:

- the angle $\theta_{i,j}$ of each i^{th} ply of each j^{th} shell laminate, grouped in θ;
- the thickness $\bar{t}_{i,j}$ of each i^{th} ply of each j^{th} shell laminate, grouped in $\mathbf{t}$;
- the height h_j of the cross section of each j^{th} beam laminate, grouped in $\mathbf{h}$;
- the width w_j of the cross section of each j^{th} beam laminate, grouped in $\mathbf{w}$.

The reinforcement fibers of the beam laminates are aligned with their longitudinal axis. For each laminate that makes up the structure, the macromechanical properties depend on the sizing design variables, on the material of each ply and the distribution of the laminates themselves

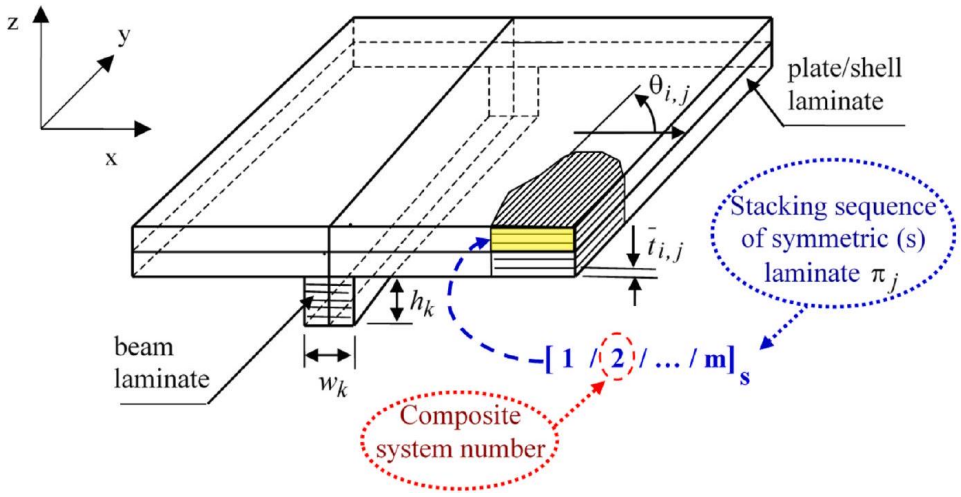


Figure 3.4: Definition of design variables for sizing and topological parts for hybrid composite structures [2]

along the structure. The material distribution for each laminate of index j is defined through variable π_j, as shown in Fig. 3.4 and is stored in vector $\boldsymbol{\pi}$, which defines the material distribution at the structure level. These variables, related to hybridization, are essential to designing optimal laminated structures [28, 33, 34].

3.3.1 Objective functions for the MOOP

The MOOP of beam-reinforced shell hybrid composite structures under static loading is formulated from the following set of objective functions [2]:

1. **weight** of the composite laminated structure:

$$W(\mathbf{x}, \boldsymbol{\pi}) = \sum_{i=1}^{n_lam} \sum_{j=1}^{n_plies(i)} \mu_{ij}(\boldsymbol{\pi})\, V_{ij}(\mathbf{x}) \tag{3.58}$$

   where

   - $\mu_{ij}(\boldsymbol{\pi})$ is the specific weight of the j^{th} ply of the i^{th} laminate;
   - $V_{ij}(\mathbf{x})$ is the volume of the j^{th} ply of the i^{th} laminate;

2. **mean value of the strain energy**, evaluated in closed form as:

$$T_1(\mathbf{x}, \boldsymbol{\pi}) = U(\mathbf{x}, \boldsymbol{\pi}) = \frac{1}{2}\, {}^t\lambda\, \bar{\mathbf{F}} \bullet {}^t\mathbf{d} \tag{3.59}$$

3. **standard deviation of the strain energy**, which describes the uncertainties in the mechanical properties of the system. The mean value and variance of the energy can be obtained as follows [2]:

$$\text{var}(U) = \sum_{i=1}^{n} S_i^2 \ \text{var}(x_i) + 2 \sum_{i \neq j=1}^{n} S_i \, S_j \ \text{cov}(x_i, x_j) = \mathbf{S}^T \ \mathbf{C}_x \ \mathbf{S} \tag{3.60}$$

where

- var(.) and cov(.) are the variance and covariance, respectively,
- $\mathbf{C}_x$ is the covariance matrix for the design variables, whose components are:

$$(\mathbf{C}_x)_{ij} = \begin{cases} \text{cov}(x_i, x_j) = \rho_{ij}\sigma_i\sigma_j, \ i \neq j, \ \rho_{ij} = \text{correlation coefficient} \\ \text{var}(x_i) = \sigma_i^2, \ i = j \end{cases} \tag{3.61}$$

- the components of the energy gradient (3.2.5) $\mathbf{S} = (S_1, \ldots, S_n)$, are defined as:

$$S_i = \left(\frac{\partial U}{\partial x_i} \right)_{\mathbf{x}^0}, \quad \mathbf{x}^0 = (x_1^0, \ldots, x_n^0) = \text{nominal values} \tag{3.62}$$

If the design variables $(x_1, \ldots, x_n)$ are uncorrelated, Eq. 3.60 is equivalent to

$$\text{var}(U) = \sum_{i=1}^{n} S_i^2 \, \text{var}(x_i) = \sum_{i=1}^{n} S_i^2 \sigma_i^2 \tag{3.63}$$

and the third objective function becomes:

$$T_2(\mathbf{x}, \boldsymbol{\pi}) = \sqrt{\text{var}(U(\mathbf{x}, \boldsymbol{\pi}))} \tag{3.64}$$

3.3.2 MOOP formulations (case studies)

Since the minimization of structural weight is paramount in the current setting (1), any definition of a MOOP must include the first objective. As a result, there exist three possible combinations of at least two of the existing three objective functions ($\sum_{k=1}^{2} {}^2\text{C}_\text{k}$):

$$\mathbf{G}_1(\mathbf{x}, \boldsymbol{\pi}) = [W(\mathbf{x}, \boldsymbol{\pi}),\ T_1(\mathbf{x}, \boldsymbol{\pi})] \tag{3.65}$$

$$\mathbf{G}_2(\mathbf{x}, \boldsymbol{\pi}) = [W(\mathbf{x}, \boldsymbol{\pi}),\ T_2(\mathbf{x}, \boldsymbol{\pi})] \tag{3.66}$$

$$\mathbf{G}_3(\mathbf{x}, \boldsymbol{\pi}) = [W(\mathbf{x}, \boldsymbol{\pi}),\ T_1(\mathbf{x}, \boldsymbol{\pi}),\ T_2(\mathbf{x}, \boldsymbol{\pi})] \tag{3.67}$$

which gives rise to two 2OOP and one 3OOP:

$$\text{Minimize}\ \ \mathbf{G}_k(\mathbf{x}, \boldsymbol{\pi}) \text{ subject to } \mathbf{g}(\mathbf{x}, \boldsymbol{\pi}) = \begin{bmatrix} 1 - \dfrac{\lambda_{crit}(\mathbf{x}, \boldsymbol{\pi})}{\bar{\lambda}_a} \\ \dfrac{d_{crit}(\mathbf{x}, \boldsymbol{\pi})}{\bar{d}_a} - 1 \end{bmatrix} \leq 0, \quad k = 1,\ 2,\ 3 \tag{3.68}$$

where $\bar{\lambda}_a$ and $\bar{d}_a$ are the allowable load factor and displacement, respectively, and whose design space is defined through the size constraints:

$$x_j^l \leq x_j \leq x_j^u, \quad j = 1, \ldots, \bar{N}_x \tag{3.69}$$

where $\bar{N}_x$ denotes the number of design variables. Moreover, the optimal solutions must satisfy the equilibrium equation (Eq. 3.20):

$$\Psi({}^t\mathbf{d}, {}^t\lambda, \mathbf{x}, \boldsymbol{\pi}) = \mathbf{R}({}^t\mathbf{d}, \mathbf{x}, \boldsymbol{\pi}) - {}^t\lambda\bar{\mathbf{F}} = 0 \tag{3.70}$$

and the arc-length equation (Eq. 3.27):

$$\Phi({}^t\mathbf{d}, {}^t\lambda, \mathbf{x}, \boldsymbol{\pi}) = 0 \tag{3.71}$$

3.3.3 Review of concepts in multi-objective optimization

A MOOP can be mathematically defined as [2]:

$$\text{Minimize } f(\mathbf{u}) = \{f_i(\mathbf{u}) : R^n \mapsto R; i = 1, \ldots, q;\ q > 1\} \text{ over } \mathbf{u},$$

$$\text{subject to the constraints} \tag{3.72}$$

$$g_j(\mathbf{u}) \leq 0;\ \{g_j(\mathbf{u}) : R^n \mapsto R; j = 1, \ldots, p\}$$
$$h_k(\mathbf{u}) = 0;\ \{h_k(\mathbf{u}) : R^n \mapsto R; k = 1, \ldots, r\}$$

Due to the fact that a MOOP usually does not have a unique solution, the present multi-objective optimization framework is built upon the concepts of dominance and Pareto optimality:

Dominance

Let $\mathbf{Q} \subseteq \mathbb{R}^n$ be the subset in the minimization problem formulated in 3.72. A solution $\mathbf{u}_1 \in \mathbf{Q}$ dominates a solution $\mathbf{u}_2 \in \mathbf{Q}$, if the evaluation of $\mathbf{u}_1$ is smaller than that of $\mathbf{u}_2$ for at least one objective and not greater for any objectives:

$$\mathbf{u}_1 \prec \mathbf{u}_2 \Leftrightarrow \begin{cases} \forall i : 1 \leq i \leq q \Rightarrow f_i(\mathbf{u}_1) \leq f_i(\mathbf{u}_2) \\ \exists j : 1 \leq j \leq q \Rightarrow f_j(\mathbf{u}_1) < f_j(\mathbf{u}_2) \end{cases} \tag{3.73}$$

where $\mathbf{u}_1 \prec \mathbf{u}_2$ denotes $\mathbf{u}_1$ dominates $\mathbf{u}_2$.

Pareto optimal design

Let $\mathbf{Q} \subseteq \mathbf{R}^n$ be the subset in the minimization problem formulated in 3.72. A solution $\mathbf{u}^* \in \mathbf{Q}$ is Pareto-optimal if and only if it is not dominated by any other solution in $\mathbf{Q}$. The set of all Pareto solutions also called the Pareto front, is represented by $\mathbf{u}^*$,

$$\mathbf{u}^* \in \mathbf{u}^* \Leftrightarrow \{\mathbf{u} \in \mathbf{Q} : \mathbf{u} \prec \mathbf{u}^*\} = \{\varnothing\} \tag{3.74}$$

Denoting the population of solutions as set $\mathbf{Q} \subseteq \mathbb{R}^n$, the individuals are sorted and ranked according to local constraint-dominance. An individual $\mathbf{u}_i \in \mathbf{Q}$ dominates an individual $\mathbf{u}_j \in \mathbf{Q}$, if any of the following conditions is verified [42]:

1. $\mathbf{u}_i$ and $\mathbf{u}_j$ are feasible, $\mathbf{u}_i$ is no worse than $\mathbf{u}_j$ for all objectives and strictly better than $\mathbf{u}_j$ for at least one objective,
2. $\mathbf{u}_i$ is feasible while individual $\mathbf{u}_j$ is not, or

3. $\mathbf{u}_i$ and $\mathbf{u}_j$ are both infeasible, but $\mathbf{u}_i$ has smaller constraint violation.

In cases where at least one of the two solutions is infeasible (conditions 2 and 3), performing an inequality check between their respective constraint violations simultaneously addresses conditions 2 and 3 (Listing 3.1).

Listing 3.1: Fortran code section that checks whether 'solution 2' dominates 'solution 1'

```
if ((viol(sol1) == 0. .and. viol(sol2) == 0. &
    .and. count(obj(:,sol2) > obj(:,sol1)) == 0 &
    .and. count(obj(:,sol2) < obj(:,sol1)) >= 1) & ! condition 1

.or. (viol(sol1) > viol(sol2))) then ! conditions 2 and 3

    ranks(sol1) = ranks(sol1) + 1 ! sol1 dominated by sol2

end if
```

The rank of any solution of index i is equivalent to the number of solutions it is dominated by plus 1. The constraint violation of an individual $\mathbf{u} = (\mathbf{x}, \boldsymbol{\pi})$ corresponds to the sum of its constraint violations [33] of the MOOP formulated in Eqs. 3.68 - 3.71:

$$\psi(\mathbf{u}) = \sum_{i=1}^{2} \delta_i(\mathbf{u}) = \sum_{i=1}^{2} \max(g_i(\mathbf{x}, \boldsymbol{\pi}),\ 0) \tag{3.75}$$

where $g_i(\mathbf{x}, \boldsymbol{\pi})$ are the inequality constraints of the problem (Eq. 3.72).

3.4 Optimization algorithm: Memetic algorithm

The MOMA, given its evolutionary nature, needs to translate each solution's genes to their equivalent phenotype (the design variables) and subsequently define the corresponding structure, including geometry definitions (from $\mathbf{x}$) and assignment of material properties (from $\boldsymbol{\pi}$). These requirements can be met through a set of functions and data structures that serve as adequate abstractions.

The mixed use of real and integer type variables in the phenotype can be dealt with as show in Listing 3.2, where `bin_2_real` is a function that performs the conversion between an array of 'bits' and the real variable it represents, while the variables encoded as integer keep the same value as the corresponding genes.

Listing 3.2: Decoding function

```
pure function decode_solution(genotype) result(phenotype)
    integer, intent(in) :: genotype(n_genes)
    real :: phenotype(n_vars)
    integer :: i
    phenotype(vars_geom) = [(bin_2_real(genotype, i), i = 1, n_vars_geom)]
    phenotype(vars_mat_seq) = genotype(genes_mat_seq)
end
```

A new `pure` function was created with the purpose of 'assembling' the structure from the phenotype of a given solution (Listing. 3.3). It performs the assignment of the solution's variables to a derived type for the layers of each laminate (Fig. 3.3), and computes and stores the structure's weight as well as other variables required by the FEM solver, such as the eccentricity and the matrix of anisotropic parameters of each ply of each laminate. The use of a derived `type` allows the definition of a streamlined interface for a `pure` function, by condensing a very complex data structure into objects for input and output.

Listing 3.3: Interface of Fortran function that defines the structure from problem variables

```
pure function get_structure_from_vars(vars) result(structure)
    real, intent(in) :: vars(:)
    type(structure_type) :: structure
end
```

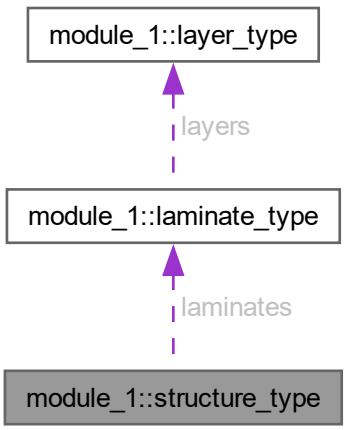


Figure 3.5: Type collaboration graph

This new approach enables the generalization of the OA, in that the analysis of any desired structure would only require the definition of a new function to assemble the corresponding structure type and a matching interface for the design variables. The `pure` keyword ensures that this function can be parallelized.

Furthermore, since the objectives of a constraint-violating solution do not impact its rank, the OA is adapted to only evaluate the third objective (standard deviation of energy, described in 3.2.5), which carries some computational cost, if its found to not violate any constraint.

The development of the MOMA is described in Chapter 4.

Chapter 4

Multi-Objective Memetic Algorithm

Since the compromise between weight/cost and stiffness/energy is a determining factor in the manufacturing of hybrid laminates, the utilization of multi-objective design procedures becomes imperative. In order to address this challenge, a Multi-Objective Memetic Algorithm (MOMA) was developed. The proposed MOMA resorts to multiple populations and learning procedures, effectively leveraging the synergy among various learning rules [2].

4.1 MOMA framework

The MOMA is characterized by a horizontal scheme of four subpopulations (SP) that evolve sequentially and in parallel with an enlarged 'virtual population' (VP) through a vertical relationship (Fig. 4.1). Each SP evolves over a set of active genes for a certain number of generations (*isolation stage*), each of which constitutes a cycle comprised of generating, evaluating and selecting solutions. Once an isolation stage is completed, a number of the best-fitted solutions, the *elite*, are migrated to the next SP (*migration stage*) (Section 4.2). The VP receives solutions from the SP during the evolutionary process and also participates in the generation of new solutions, as further described in Section 4.3.

Each completion of the sequential evolution between SP_1 and SP_4 constitutes an *epoch*, which is the longest unit of evolutionary time of the algorithm.The *success rate* of the operators during an epoch dictates their chance of being selected at every generation of each SP's *isolation stage* for the next epoch, drawing inspiration from Dawkins' interpretation of classical Darwinism, according to which genetic and cultural evolution occur simultaneously [2] (Section 4.4).

Algorithm 1 presents the general layout of the MOMA.

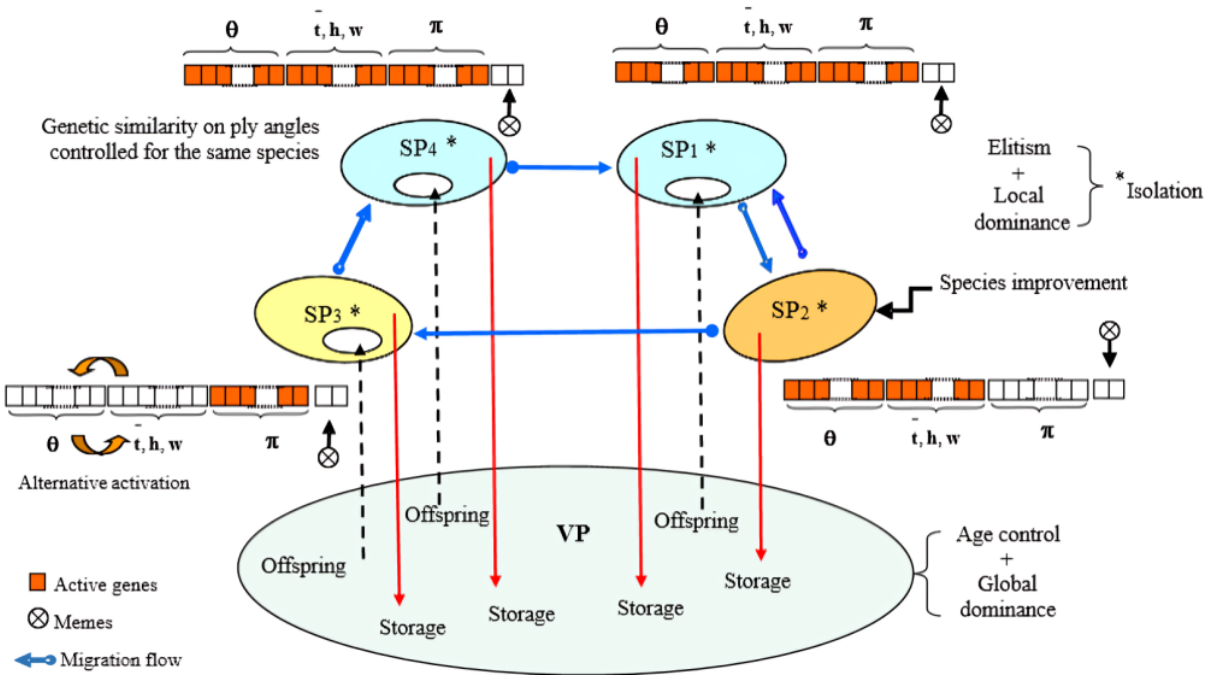


Figure 4.1: Proposed structure for the Multi-Objective Memetic Algorithm (MOMA)

Algorithm 1 Multi-Objective Memetic Algorithm (MOMA)

for each *epoch* **do**
  for each SP **do**
    Migrate the elite of the previous SP to the current one
    for each *generation* **do**
      Update the state of active/non-active genes (4.2)
      Select crossover and mutation operators (4.4)
      Generate solutions through the selected mutation operator (4.5)
      Generate solutions through the selected crossover operator (4.6)
      Remove solutions that are too similar to existing ones (4.7)
      Generate random solutions to replace the removed solutions (4.5.1)
      Evaluate all newly generated solutions through the FEM solver (3.2)
      Assign fitness to each solution in SP (4.2.1)
      Update success events (4.4)
      Update the VP (4.3)
    end for
  end for
end for

4.2 Design space decomposition: Subpopulations (SP)

Understandably, the objective functions and constraints defined for the MOOP should remain unaltered throughout the optimization process. Nonetheless, changing the set of design variables (3.3) that are active at a given stage leads to a decomposition of the design space, which can enhance exploration [28] (Fig. 4.1). The distinct implications of each set of design variables justify a natural decomposition of the MOOP (3.3.2) (Fig. 3.1):

- the ply thickness variables, the dimensions of the beam cross sections and the material distribution influence the weight of the structure;
- the fiber orientations of the shell laminates' plies are associated with anisotropy;
- the stacking sequence at the laminate level and the distribution of laminates along the structure are associated with the topology of the optimization problem.

Considering the previous definitions, the following decomposition is adopted:

- SP_1: all genes are active during the isolation stage;
- SP_2: the genetic segment corresponding to the material sequences of the laminates are inactive;
- SP_3: the genetic segments corresponding to the shell laminates' ply angles and the remaining geometric variables are activated in an alternated manner in each generation;
- SP_4: all genes are active.

The design variables define the phenotype of a solution:

$$\mathbf{u} = (\mathbf{x}, \boldsymbol{\pi}) \tag{4.1}$$

which has direct correspondence to a specific genotype:

$$\overline{\mathbf{u}} = (\bar{\mathbf{x}}, \bar{\boldsymbol{\pi}}) \tag{4.2}$$

where $\bar{\mathbf{x}}$ and $\bar{\boldsymbol{\pi}}$ represent the genetic segments that correspond to the phenotypes of the sizing variables, $\mathbf{x} = (\theta, \mathbf{t}, \mathbf{w}, \mathbf{h})$, and topology variables, $\boldsymbol{\pi}$, respectively.

The presence of a group of integer-type variables, $\boldsymbol{\pi}$, which determine the material distribution along the structure, originates significant discontinuities in the search space. This phenomenon

can be adequately dealt with by leveraging the concept of *species* (Definition 4.2.1) to identify continuous sub-regions of the search space and consequently increase the adaptability of the optimization process in the exploration of local *optima* [28]. Rules based on the Species Conservation paradigm are imposed during migration to prevent the overrepresentation of some species and avoid the premature elimination of others [28].

Definition 4.2.1 (Species)**.** *All solutions* $\mathbf{u}$ *with an equal segment* $\boldsymbol{\pi}$ *of the phenotype, which means that they possess the same material on the stacking sequence of each laminate and the same distribution of these laminates on the composite structure, belong to the same species. (Fig. 3.4)* [28].

4.2.1 Local fitness assignment

The local fitness assignment, which is presently extended for an arbitrary number of objectives, not only depends on local dominance evaluation (3.3.3) but also applies the concept of *niching* [43]. This procedure aims to penalize same-rank solutions whose objectives are comparatively closer and therefore do not provide new, sufficiently distinct compromises between the objectives being considered, while benefiting those which provide greater exploratory potential and possess the same exploitative effect. This effect is achieved by calculating the average fitness of each rank and applying a bias to obtain the fitness of each solution belonging to that rank, such that the rank's average fitness remains the same [42].

Assuming that 'rank' stores the ranks of all solutions in the current SP, which are computed as shown in Listing 3.1, and N = size(SP), the average fitness of rank k is given by:

$$\text{avg_fitness}(k) = N - \text{count}(\text{rank} < k) - \frac{\text{count}(\text{rank} == k) - 1}{2} \tag{4.3}$$

If 'obj' is a n_{obj} x $n_{solutions}$ array such that obj(i) returns the objectives of solution i, the niche count of solution i is

$$\begin{aligned} \text{niche_count} &= \sum_{\substack{j=1 \\ rank(j)=k}}^{N} \max\left(1 - \sqrt{\frac{\text{normalized_distance}(i,\,j)}{\text{reference_distance}}},\ 0\right) \\ \text{normalized_distance}(i,\,j) &= \left\| \frac{\text{obj}(i) - \text{obj}(j)}{\Delta\text{obj(VP)}} \right\| \\ \text{reference_distance} &= \frac{\text{sum}(\Delta\text{obj(VP)})}{N - 1} \end{aligned} \tag{4.4}$$

where Δobj(VP) is the difference between the maximum and minimum values of each objective function present in VP:

$$\Delta\text{obj(VP)} = \max(\text{obj(VP)}) - \min(\text{obj(VP)}) \tag{4.5}$$

A higher niche count will lead to a decreased shared fitness:

$$\text{shared_fitness}(i) = \frac{\text{avg_fitness}(k)}{\text{niche_count}(i)} \tag{4.6}$$

and a lower overall fitness:

$$\text{fitness}(i) = \text{avg_fitness}(k) \cdot \text{count}(\text{rank} == k) \frac{\text{shared_fitness}(i)}{\sum\limits_{\substack{j=1 \\ rank(j)=k}}^{N} \text{shared_fitness}(j)} \tag{4.7}$$

Fig. 4.2 illustrates the isolation stage.

4.3 Evolution history: the Virtual Population (VP)

The evolutionary framework described for the SPs implies continuous generation and removal of solutions at a local level. The underlying mechanisms that exist to ensure that diversity is preserved to some extent, namely the genetic similarity control and the constraints imposed during migration, may result in the removal of moderately virtuous solutions that could help improve global search. On the other hand, these control mechanisms require that these populations have a limited size, otherwise there would be no criterion for the removal of the worst solutions from the populations to prevent them from taking part in the recombination schemes (crossover) and waste computational resources. Thus the need arises to both store some desirable solutions that would be lost along the evolutionary process and, simultaneously, all non-dominated solutions, in order to build the global Pareto front.

The VP is an extended population that is continuously updated along the entire evolutionary process and, in contrast to the SPs, has a dynamic size. At the end of each generation of each SP, the newly-generated solutions that meet a given fitness threshold are stored in the VP while preserving their instances in SP_i, therefore acquiring parallel existence. After each update, dominance evaluation is performed, as described in 3.3.3, resulting in an updated global Pareto front at the end of each generation.

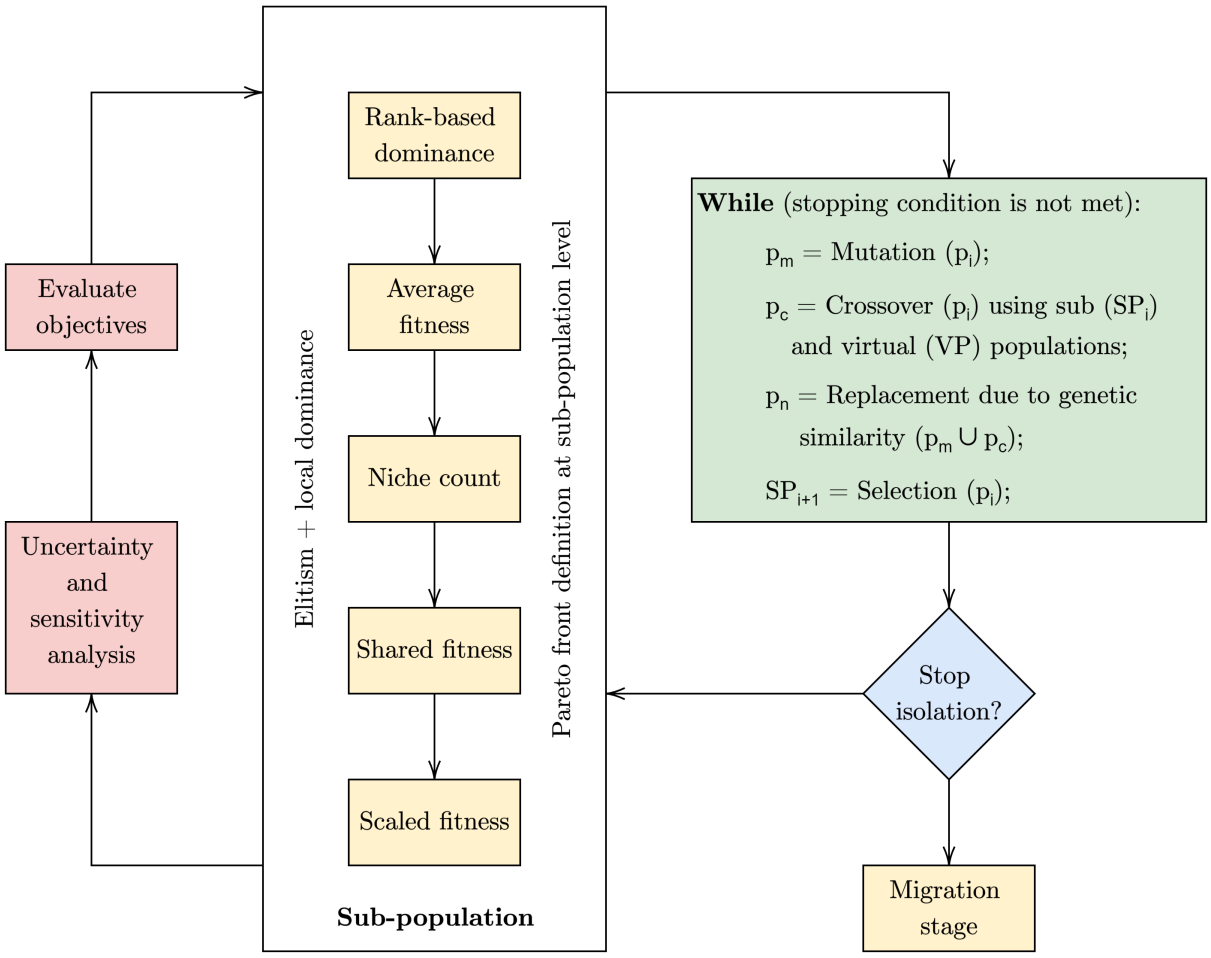


Figure 4.2: Fitness assignment and local evolutionary scheme during the isolation stage

Since, as evolution progresses, many of these solutions that had acquired 'virtual' status will inevitably become comparatively worse and lose any exploratory advantage they might have provided, a rule to remove them from the VP becomes necessary. The adopted rule relies on two metrics, the rank and the age of each solution [2], which are both updated at the end of each generation, with the ranks computed as shown in 3.3.3 and the age of each solution being incremented at each generation from the moment they enter the VP. The age metric allows for the definition of a lifespan during which dominated solutions may remain in the VP that does not rely on constraining the population size, since the VP is dynamically sized. Those solutions that reach the *lethal age* and are dominated (rank > 1) will be removed from the VP, while those that belong to the Pareto front (rank = 1) will reside in the VP for as long as they keep their non-dominated status. This mechanism can be implemented as described in Listing 4.1. This implementation relies on initializing the ages to -1 in order to control whether a given solution in SP_i was already stored in the VP; if not, it can be added, depending on its fitness, and its age becomes 0. The population arrays store the indices of the solutions that inhabit them, and

`dominance` relies on the comparison show in Listing 3.1 to compute the rank of each solution in the VP.

Listing 4.1: Dynamic update of the VP

```
VP = [VP, pack(SP, solution_age(SP) < 0 .and. solution_fit(SP) >= min_fit)]
solution_age(VP) = solution_age(VP) + 1
VP = pack(VP, solution_rank(VP) == 1 .or. solution_age(VP) < lethal_age)
solution_rank(VP) = dominance(sol_indices=VP)
```

As was previously hinted, the VP does not exist solely for the purpose of recording the Pareto front. The enlarged storage it provides, which includes both the Pareto-optimal as well as dominated but otherwise reasonably fitted solutions, can be leveraged by some crossover operators that draw parents from this pool in order to generate new offspring, as discussed in Section 4.6. The described vertical relationship whereby offspring generated from the VP are inserted in SP_i augments the global search beyond using the SPs in isolation and is an example of Lamarckian learning [34].

4.4 Memes

In the present MOMA, solutions can be generated by several different operators, which can be divided in two categories: crossover (4.6) and mutation (4.5). Correspondingly, each statically-sized SP possesses an immutable layout where, apart from the elite, two groups are reserved for the solutions that are to be generated by one operator of each type. The two mutation operators of the MOMA are alternatively selected each generation, since Implicit Mutation is required to assure diversity throughout the epochs [2]. The cultural evolution feature of MOMA manifests itself in the adaptive selection probability of the n crossover operators, whereby, for a given SP of index p, the success rate of crossover operator k at epoch $t-1$ determines its probability $P_k(t,p)$ of being selected at each generation of epoch t:

$$P_k(t,p) = \frac{\text{success_rate}_k(t-1,p)}{\sum_{i=1}^{n} \text{success_rate}_i(t-1,p)} \tag{4.8}$$

The success rate is defined as:

$$\text{success_rate}_k(t,p) = \frac{\text{n_success_events}_k(t,p)}{\text{n_events}_k(t,p)} \tag{4.9}$$

A success event occurs when a newly-generated solution corresponds to a fitness greater than the minimum fitness found in that generation's elite as evaluated in that same generation, and the number of events corresponds to the total number of solutions generated by a given operator in a given epoch for a given SP. During the first epoch, given that no information is known about each operator's success rate, the same selection probability is attributed to all crossover operators. In order to prevent a crossover operator from never being selected again if its success rate ever becomes 0, and to stabilize the selection process in general, $P_k(t, p)$ must be constrained to a range:

$$P_k(t, p)' = \min(\max(P_k(t, p),\ P_{min}(p)),\ P_{max}(p)) \tag{4.10}$$

Presently, $P_{min}(p)$ depends on the number of crossover operators active in SP of index p, n_crossover_operators(p), and is defined as

$$P_{min}(p) = \frac{1}{\text{n_crossover_operators}(p) + 1} \tag{4.11}$$

The probability corrections added to the corrected operators are taken from the operators whose probability is above that threshold, automatically setting $P_{max}(p)$.

The operator that generates each solution is represented by its *meme* (Table 4.1).

Table 4.1: Operator-*meme* correspondence

Operator	Code
Memes for crossover group	
Elitist parameterized Uniform Crossover (EpUC) (4.6.1)	1
Elitist Hybrid Crossover with genetic improvement (EHCgi) (4.6.2)	2
Age-Dominance parameterized Uniform Crossover (ADpUC) (4.6.3)	3
Age-Dominance parameterized Selfish Gene crossover (ADpSG) (4.6.4)	4
Memes for mutation group	
Implicit Mutation (4.5.1)	5
Controlled Mutation (4.5.2)	6
Memes for substitutions	
Implicit Mutation - newly-generated solution	7
Implicit Mutation - clone replacement	8
Implicit Mutation - similarity replacement	9

4.5 Mutation operators

Mutation operators can have multiple formulations and purposes. They may be used to generate completely random solutions and therefore have an exploratory effect, or be exploitative in nature, if they aim to generate a potentially better-fit solution from an existing one. These operators can be classified in terms of their *arity* as either *nullary* (if they take no solutions as argument) or *unary* (if they take a single solution as argument). The MOMA incorporates two mutation operators, namely Implicit Mutation (4.5.1) and Controlled Mutation (4.5.2).

4.5.1 Implicit Mutation

The Implicit Mutation operator generates completely random solutions, which serves to both create an initial population for the evolutionary process, as well as to continually introduce new solutions so that the optimization process does not get stuck in local minima. In biological terms, this process guarantees the diversity of the population [44] (Algorithm 2).

Algorithm 2 Implicit Mutation

for each *gene* **do**
    offspring(*gene*) ← random_int(gene_bounds(*gene*,:))
end for

4.5.2 Controlled Mutation

The Controlled Mutation operator aims to incorporate data related to the behavior of state variables of the structural system, by defining a relationship between the stress field in the composite structure and the corresponding genotype [28] (Algorithm 3).

Algorithm 3 Controlled Mutation

Randomly select a solution from the *elite*.
Rank each point on the corresponding structure by decreasing order of the stress constraints.
Identify the sets of variables associated with the top (m_{sup}) and bottom (m_{inf}) stress constraints.
if the stress constraints of the top group are violated **then**
    Mutate the corresponding genes, aiming for the non-violation of the constraints.
end if
if the stress constraints of the bottom group are not violated **then**
    Mutate the corresponding genes, aiming for weight minimization.
end if

4.6 Crossover operators

Contrary to mutation operators, described in the previous section, which utilize at most one parent to generate a new solution, crossover operators, by definition, involve the presence of multiple parents, which share their genetic material in order to generate offspring. Consequently, any crossover operator is at least (*binary*). The several crossover operators described in this section mainly differ on how the parents are selected, nevertheless, when it comes to the actual crossover process, they resort to the same algorithm, the Uniform Crossover [1]. It selects, for each gene of the offspring, the corresponding allele from one of the parents, with equal probability (Algorithm 4).

Algorithm 4 Uniform Crossover

for each *gene* **do**
    offspring(*gene*) ← parents_genes(*gene*,ceiling(Unif(0,1)*2))
end for

In order to comply with the design-space decomposition described in Section 4.1, the resulting offspring must not introduce new variations of the genetic segments defined as *non-active* at a given point in the evolutionary process (4.2). Therefore, it must inherit *all* alleles corresponding to the current non-active segments (if there are any) from one of the parents. The selection of the parent that 'donates' these segments is dependent on the crossover operator being used.

4.6.1 Elitist parameterized Uniform Crossover (EpUC)

This crossover operator employs the concept of *elitism*, since it selects two parents according to a probability defined by their fitness. The current SP_i is sorted by fitness and divided into two groups, the *elite* and the *worst*. One parent is randomly selected from each group under a probability proportional to their fitness [1]. The parent from which the offspring inherits the non-active genes is randomly selected considering equal probability (Algorithm 5).

Algorithm 5 Elitist parameterized Uniform Crossover (EpUC)

parents_genes(:,1) ← fitness-weighted random selection from the elite
parents_genes(:,2) ← fitness-weighted random selection from the worst
offspring ← uniform_crossover(parents_genes) (Algorithm 4)
offspring(non_active) ← parents_genes(non_active,ceiling(Unif(0,1)*2))

4.6.2 Elitist hybrid crossover with genetic improvement (EHCgi)

This crossover operator shares the same parent selection process as the previously described one (4.6.1), however, it applies the Uniform Crossover (Algorithm 4) several times to the selected parents in order to generate several potential offspring [41]. The non-active genes of each local solution i are inherited from the parent closest to it as determined by the Hamming distance applied to the active genes:

$$\mathrm{dist}(\mathrm{x}, \mathrm{y}) = \sum_{i=1}^{\mathrm{size(x)}} |x_i - y_i| \tag{4.12}$$

The local search procedure requires that one offspring is selected to then be included in SP_i. A new offspring selection mechanism is proposed, which makes use of the concept of constraint-based dominance described in 3.3.3. This provides a solution to the problem of defining an adequate fitness function through a linear combination of the MOOP objectives (3.3.1). It is nonetheless necessary to either evaluate or approximate each solution's objectives, as described further.

Approximations for the objectives and constraint violation

For each local solution, the weight of the corresponding structure, which constitutes the first objective of the MOOP (Eq. 3.58), can be obtained directly from its phenotype (Listing 3.3). On the other hand, obtaining the actual values for the energy (Eq. 3.59) and its standard deviation (Eq. 3.60), as well as the constraint violations (Eq. 3.68) would require evaluating the corresponding structure via the FEM solver, which would be too computationally expensive [41], especially given that this crossover operator might generate tens or even hundreds of offspring candidates; therefore, a set of approximations is used. Recalling the definition for the energy gradient (3.2.5), the strain energy of the structure corresponding to the offspring candidate with phenotype $\mathbf{z}_k$ can be approximated as [41]:

$$\tilde{U}(\mathrm{z}_k) = \tilde{U}(\mathrm{x}_0) + \sum_{j=1}^{N_{var}} \frac{\partial \tilde{U}(\mathrm{x}_0)}{\partial x_j}(z_{k,j} - x_{0,j}) \tag{4.13}$$

where N_{var} is the number of active design variables and $\mathbf{x}_0$ corresponds to the phenotype of the closest parent. The estimate for the constraint violation is given by the distance (Eq. 4.12) between the offspring candidate and each of its parents, as well as their constraint violations, $\mathrm{viol}(p_1)$ and $\mathrm{viol}(p_2)$:

$$\text{viol}(z_k) = \frac{\text{viol}(p_1) \cdot \text{dist}(z_k, p_2) + \text{viol}(p_2) \cdot \text{dist}(z_k, p_1)}{\text{dist}(z_k, p_1) + \text{dist}(z_k, p_2)} \tag{4.14}$$

Algorithm 6 exemplifies an implementation of this operator. Note that the evaluation of a local solution's objectives is only required if its constraint violation approximation is 0, since they will not impact its rank otherwise (all constraint-violating solutions are dominated by non-constraint violating ones and sorted in decreasing rank with increasing constraint violation), which constitutes another advantage when compared to the use of a fitness function. More importantly, this condition is in fact required by the performance-enhancing update presented in 3.4, since an *approximately* constraint-violating local solution will necessarily have a constraint-violating parent which itself will not have all of its objectives evaluated and the attempt at approximating these objectives will likely lead to an execution error.

Algorithm 6 Elitist Hybrid Crossover with genetic improvement (EHCgi)

```
parents_genes(:,1) ← fitness-weighted random selection from the elite
parents_genes(:,2) ← fitness-weighted random selection from the worst
for each local solution i do
    offspring(:,i) ← uniform_crossover(parents_genes) (Algorithm 4)
    dist(1) ← sum(abs(offspring(active,i) - parents_genes(active,1)))
    dist(2) ← sum(abs(offspring(active,i) - parents_genes(active,2)))
    p ← minloc(dist, dim=1)                                  ▷ closest parent
    offspring(non_active,i) ← parents_genes(non_active,p)
    viol(i) ← (viol(1) · dist(2) + viol(2) · dist(1))/sum(dist) (Eq. 4.14)
    if viol(i) == 0 then
        vars(i) ← decode_solution(offspring(:,i)) (Listing 3.2)
        obj(1,i) ← weight of the corresponding structure (Eq. 3.58)
        obj(2,i) ← obj(2,p) + dot(energy_grad(p), vars(i) - vars(p)) (Eq. 4.13)
        obj(3,i) ← obj(3,p) + dot(energy_grad(p), energy_grad(p))
    end if
end for
Rank the offspring candidates (3.3.3)
Return the first solution whose rank is equal to 1
```

Extensions to dominance evaluation

This operator originally resorted to a local fitness function to evaluate the n_{local} solutions generated from the two selected parents, corresponding to a linear combination of each solution's objectives and constraint violations [41], which was replaced with a local dominance evaluation in order to effectively deal with an arbitrary set of objectives. Nonetheless, there was opportunity for improvement since this first iteration does not consider the existing solutions in SP_i. The practical consequence of this is that, after ranking these n_{local} solutions, the

crossover operator, by default, chooses the first one that corresponds to a rank of 1, out of potentially several non-dominated solutions, without being able to further discriminate between these *locally-optimal* solutions. It can be inferred that some of these *locally-optimal* solutions could indeed end up on the Pareto front when placed in SP_i, while others would be dominated by existing solutions (alternatively, it could be that none of the local solutions would be non-dominated in SP_i). By including the existing solutions in SP_i in the dominance evaluation, it is possible to obtain a smaller set of local solutions that correspond to the minimum rank - note that it is no longer assured that any of them will be of rank 1.

In order to implement this feature, the EHCgi crossover was further updated to, depending on the user's input, assemble arrays for the objectives and constraint violations of both the local (temporary) solutions and the solutions belonging to SP_i, which are then passed to the dominance evaluation function. Furthermore, a similar process is implemented where the solutions in the VP are considered instead of those in SP_i, leading to three types of dominance evaluation to choose from. This implementation is facilitated by Fortran's `optional` attribute and `present` intrinsic. It should be added that the version of dominance evaluation that is used persists for all calls to this crossover operator, determined by a parameter. Nonetheless, it would be possible to introduce all procedures in the same program execution, as variants of meme 2.

4.6.3 Age-Dominance parameterized Uniform crossover (ADpUC)

Contrary to the previously presented crossover operators, which perform parent selection on the current SP_i, the Age-Dominance parameterized Uniform Crossover [28] selects two parents from the VP. The selection of the first parent follow the heuristic that the maturity/potentiality of the solutions belonging to the VP is dependent on a normal distribution of their ages. The second parent is selected from a group belonging to VP under a certain rank threshold considering a uniform probability distribution [28].

Considering the cumulative normal distribution function [45]:

$$P_0 = \Phi(\beta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\beta} e^{-z^2/2} \mathrm{d}z \tag{4.15}$$

its inverse, the quantile function known as *probit*, can be approximated, with an error less than 10^{-5}, as [46]:

$$\beta_p = r - \frac{a_1 + a_2 r + a_3 r^2}{1 + a_4 r + a_5 r^2 + a_6 r^3}, \quad r = \sqrt{\ln\left(\frac{1}{(1-P_0)^2}\right)}, \quad \text{a} = \begin{bmatrix} 2.515517 \\ 0.802853 \\ 0.010328 \\ 1.432788 \\ 0.189269 \\ 0.001308 \end{bmatrix} \tag{4.16}$$

Given a probability P_0, the first parent corresponds to index p_1 (Eq. 4.17) in the age-sorted VP of size N:

$$p_1 = (\beta - z_{inf})\frac{N-1}{z_{sup} - z_{inf}} + 1, \quad z_{inf} = -3, \quad z_{sup} = 3 \tag{4.17}$$

Algorithm 7 Age-Dominance parameterized Uniform Crossover (ADpUC)

parents_genes(:,1) ← normal random selection from VP considering Eqs. 4.16, 4.17
parents_genes(:,2) ← uniform random selection from VP(rank < threshold)
offspring ← uniform_crossover(parents_genes) (Algorithm 4)
offspring(non_active) ← parents_genes(non_active,ceiling(Unif(0,1)*2))

4.6.4 Age-Dominance parameterized Selfish Gene crossover (ADpSG)

Similar to ADpUC (4.6.3), the Age-Dominance parameterized Selfish Gene crossover (ADpSG) randomly selects the first parent from the VP according to a normal distribution of age. However, it generates the second parent by directly applying the Selfish Gene theory [10], according to which a population is a pool of genes, and the individual genes fight for the same spot of the chromosome in the genotype [2]. In SG theory, the gene becomes the most important unit of evolution and its replication is encouraged.

The segments corresponding to the inactive genes are inherited from the first parent. In Algorithm 8, p_2 is a pseudo-parent for the mating procedure, also known as *stud* [34]. The stud is generated by evaluating the *performance measure*, PM, of each allele by weighting the frequencies of alleles in the VP by the corresponding rank of each individual.

Listing 4.2: Stud generation.

```
do locus = 1, n_genes
    do j = 1, n_alleles(locus)
        has_allele(:) = solution_genes(locus,VP) == alleles(locus,j)
        PM(j) = sum(1./pack(ranks, has_allele))
    end do
    stud(locus) = alleles(locus,maxloc(PM(:n_alleles(locus)), dim=1))
end do
```

Algorithm 8 Age-Dominance parameterized Selfish Gene crossover (ADpSG)

parents_genes(:,1) ← normal random selection from VP considering Eqs. 4.16, 4.17
parents_genes(:,2) ← generate_stud() (Listing 4.2)
offspring ← uniform_crossover(parents_genes) (Algorithm 4)
offspring(non_active) ← parents_genes(non_active,1)

4.7 Similarity control and evolution history

In order to ensure genetic diversity and therefore good exploration of the design space, the OA keeps track of every solution generated during the program's execution, storing its genome, phenotype (design variables) and objective function evaluations and constraint violations. Each time a new solution is generated, its genetic material is compared to that of all previously generated solutions and the solution is subsequently eliminated if it is found to be a clone. This process guarantees that each solution is unique and that there is an exact correspondence between a given solution index and a specific genome. As a result, if and when a solution is eliminated from the SP where it resides at a given point in time, its genetic material can no longer reappear further along the evolutionary process. Furthermore, if a given solution ever enters the VP (by means of its current local fitness meeting a certain threshold that corresponds to a multiple of the minimum fitness value of the *elite* (4.3)), and is later eliminated from it, that same solution will no longer have a chance of (re)entering the VP. This design choice is based on the premise that, if the presence of that solution in the VP was previously deemed inadequate - by virtue of it being sufficiently comparatively worse than some of the remaining solutions - it should not be expected that its fitness will ever be competitive again.

Chapter 5

Improving the efficiency of the Finite Element program

The highly computationally-intensive nature of the nonlinear finite element solver used to evaluate the solutions meant that potential optimizations of the program would allow for a higher number of executions of the memetic optimizer and therefore increase the iterative potential of the research to be conducted. With this in mind, several opportunities for the improvement of the efficiency of the program were identified and their implementations are described in this chapter.

5.1 General aspects of the original algorithm

5.1.1 Read/write operations

The original finite element solver was developed in the early nineties [36] and the underlying circumstances dictated several design choices, some of which do not exactly align with the capabilities of modern computers. The main focus was on saving computer memory (RAM), which was often the main computational constraint at the time, and therefore the original program was designed to heavily rely on `read`/`write` operations, namely of some matrices that were computed as intermediate results. Although this process of reading and writing variables to and from files eases the demand on computer memory, it is substantially slower than storing and accessing values to and from the random-access memory.

Stiffness matrix computation

Most notably, the several components of the element strain matrices ($\mathbf{B}^e$) (Eqs. 3.18, 3.19) of each element were originally sequentially written to an `unformatted` file and later read when computing each $\mathbf{K}^e$. The approach taken to improve speed of execution involved storing their values in global variables instead.

Solution method

The original program resorted to a frontal solver, a technique developed by Bruce Irons [47] to efficiently solve sparse linear systems by avoiding a large number of operations that involve zero terms. One additional advantage that this technique provided at that time was that it allowed the individual element stiffness matrices ($\mathbf{K}^e$) to be read from files as needed so that the whole sparse stiffness matrix was never explicitly created. Given that this memory constraint is no longer relevant, the frontal solver method was replaced with a function that assembles the whole system, using the element matrices stored in variables, which is then solved using the Gauss elimination method. The results obtained with the new implementation were shown to be equivalent to those obtained with the old method and the Gauss elimination in particular was benchmarked against the LAPACK library [48] implementation provided by the Intel® Math Kernel Library [49]

5.2 Requirements for parallelism

A defining aspect of the original program was the extensive use of non-pure *subroutines* as basic units of the code structure. One requirement for `do loop` concurrency is the absence of side-effect inducing statements, which necessarily includes non-pure procedures. Among the several subroutines replaced with equivalent `pure` functions are the ones for obtaining the element stiffness matrices, which can now be computed in parallel; this change carries significant importance since the CPU workload, in the context of the current spherical shell structure problem, is roughly evenly divided between the computation of shell element stiffness matrices and solving the system $\mathbf{d} = \mathbf{K}^{-1}\,\mathbf{f}$. Moreover, since the new Gauss elimination implementation builds the entire global stiffness matrix, which is directly used without resorting to intermediate `read` operations, each inner loop of the elimination is independent and can be carried out in parallel. The `do concurrent` construct, from the Fortran 2008 standard, may be used to assert that a loop is safe for parallel execution [50] (Listing 5.1).

Listing 5.1: Forward elimination using the `do concurrent` language construct

```
do i = 1, n - 1
    factor(i:) = 1./A(i,i)*A(i,i:)
    do concurrent (j = i + 1 : n)
        A(j,i:) = A(j,i:) - A(j,i)*factor(i:)
    end do
end do
```

5.3 The FEM solver as part of the optimization algorithm

Apart from the aforementioned changes that were introduced to improve the efficiency of the FEM solver itself, the fact that it belongs to a larger scope which is the OA means that some adaptations could be made that would not necessarily make sense in the context of a standalone FEM program. Given that the broader algorithm is intended to optimize a specific structure with well defined characteristics within a given design space, not every constant needs to be evaluated every time a new solution is generated. One such example is the spatial distribution of the nodal coordinates of the structure, which is the sole factor determining the values of the linear components of the elemental strain matrices ($\mathbf{B}^e$). As a result, the layout of the program was changed such that, at the beginning of the optimization process, these matrices, as well as their transposes, are computed and stored for the lifetime of the program.

Furthermore, the original implementation of the program performed the evaluation of the equivalent stiffness matrices of the laminate ($\mathbf{D}$) attributed to each element on every iteration of the displacements, regardless of whether or not at least one of the plies of each laminate were determined to have undergone yielding. Consequently, the functions pertaining to the computation of the element stiffness matrices ($\mathbf{K}$) were updated to check for the occurrence of yielding and only in such case perform the computation of the equivalent elastoplastic stiffness matrix, $\mathbf{D}_{ep}$.

Chapter 6

Applications

This chapter discusses applications of the MOMA previously described in Chapter 4 to the structural optimization of a spherical composite shell structure with beam reinforcements, as shown in Fig. 6.1. The shell is hinged along its perimeter and a point load F_{max} = 2kN acts on its central node. The structure is composed of three shell laminates (indices 1 to 3) and three beam laminates (indices 4 to 6), plotted as bold lines.

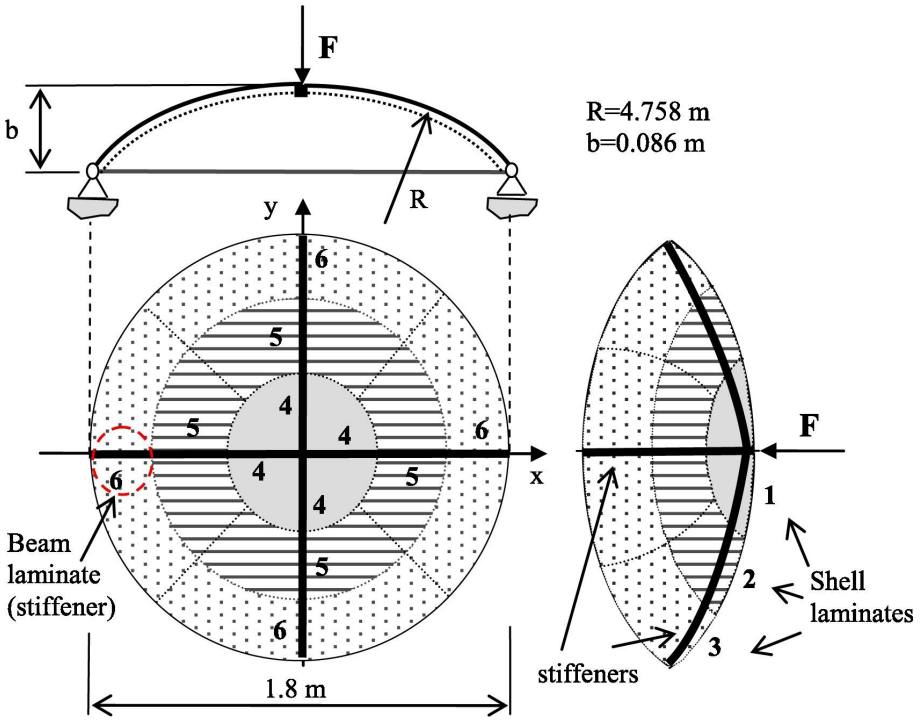


Figure 6.1: Spherical shell structure and laminate definition. The beam laminate stiffeners are displayed in bold [2].

Given the axial symmetry of the structure and in order to reduce the computational expense

of its evaluation, the corresponding finite element model was modelled for one fourth of the structure, which meant properly defining its boundary conditions to accurately represent the connections to the non-modelled part of the structure, and dividing the total applied load by 4 (Fig. 6.2). Consequently, the values of the objective functions that are later presented in this chapter correspond to one-fourth of the structure.

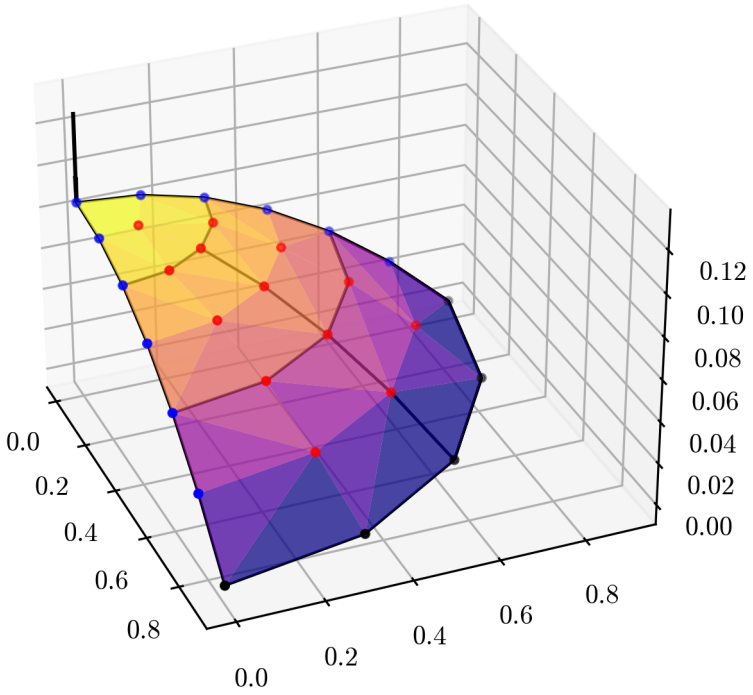


Figure 6.2: Boundary conditions for $\frac{1}{4}$ of the spherical shell composite structure. Nodes that are colored blue have prescribed displacements and rotations, black nodes represent hinges. A point load is applied to the center node (0,0).

As far as material selection is concerned, the present MOMA considers four different composite systems [2], namely one carbon/epoxy, two glass/epoxy and one Kevlar/epoxy composites [40], whose properties can be found in Table 6.1.

Table 6.1: Mechanical properties of the composite materials used in the laminates

Composite material	Index	E_1	E_2	G_{12}	ν	X	Y	S	ρ
		[GPa]			[-]	[MPa]			[kg/m^3]
CFRP: T300/N5208	1	181.0	10.3	7.17	0.28	1500	40	68	1600
GFRP: Scotchply 1002	2	38.60	8.27	4.14	0.26	1062	31	72	1800
GFRP: E-glass/epoxy	3	43.00	8.90	4.50	0.27	1280	49	69	2000
KFRP: Kevlar 49/epoxy	4	76.00	5.50	2.30	0.34	1400	12	34	1460

Recalling from Chapter 4 that the material selection process occurs for each ply of each

laminate, it is sensible to define the set of different material sequences that a given laminate may display, due to manufacturing constraints on the one hand, and design experience regarding material properties on the other hand [2]. Subsequently, it shall be imposed that the Kevlar/epoxy composite system may only be chosen for the inner plies of the (symmetric) shell laminates [2] due to their shear strength, which, in addition, must contain at least two different composite systems, given the hybrid nature of the composite structures that are the subject of this thesis. Given these constraints, there are 33 possible stacking sequences of these four materials for each shell laminate (Table 6.2). Furthermore, the composite system of index 2 is prescribed for all six plies of the beam laminates, owing to its shear strength [2].

Table 6.2: Material sequences

allele	π_1	π_2	π_3	allele	π_1	π_2	π_3	allele	π_1	π_2	π_3
1	1	1	2	12	2	2	1	23	3	3	1
2	1	1	3	13	2	2	3	24	3	3	2
3	1	2	1	14	2	3	1	25	1	1	4
4	1	2	2	15	2	3	2	26	2	2	4
5	1	2	3	16	2	3	3	27	3	3	4
6	1	3	1	17	3	1	1	28	1	2	4
7	1	3	2	18	3	1	2	29	2	1	4
8	1	3	3	19	3	1	3	30	1	3	4
9	2	1	1	20	3	2	1	31	3	1	4
10	2	1	2	21	3	2	2	32	2	3	4
11	2	1	3	22	3	2	3	33	3	2	4

The allowable critical displacement in buckling or FPF modes, defined in Eq. 3.45, is $\bar{d}_a = 9.0\times10^{-2}$m, while the allowable load factor, defined in Eq. 3.43, is $\bar{\lambda}_a = 1.0$. The size constraints on the design variables are:

$$\begin{gathered} -90^\circ \leq \theta_{i,j} \leq 90^\circ \\ 1.0\times10^{-3}\text{m} \leq t_{i,j} \leq 4.0\times10^{-3}\text{m} \\ 2.0\times10^{-2}\text{m} \leq h_j \leq 4.0\times10^{-2}\text{m} \\ 5.0\times10^{-3}\text{m} \leq w_j \leq 2.0\times10^{-2}\text{m} \end{gathered} \tag{6.1}$$

The size constraints are automatically enforced through the discretization of each variable into its respective genetic encoding. Give that each real-type variable is encoded with 4 bits, their

respective ranges are discretized in 16 intervals.

The standard deviations $\sigma(\theta_i) = 5°$ are prescribed for all variables belonging to the ply angle group, θ, and $\sigma(\mu_i) = 0.05\mu_i$ is considered for each of the remaining design sizing variables.

Previous results obtained by Conceição António [2] illustrate the expected trade-off between both weight and energy minimization (Eq. 3.65) and weight and energy standard deviation minimization (Eq. 3.66). Since the study of the 3OOP (Eq. 3.67) is of considerable relevance to the design of hybrid composite structures, it becomes necessary to evaluate the degree to which the aforementioned trade-offs persist when all three objectives are considered and, more importantly, what type of trade-off exits between the minimization of the strain energy and its standard deviation.

The parameters used for the MOMA can be found in Table 6.3.

Table 6.3: Parameter definitions for the MOMA

General parameters	
n epochs	30
n digits for binary coding of variables of type real	4
n digits for integer coding of variables of type integer	3
Subpopulations	
Population size	27
Elite group size	11
Mutation group size	5
n generations per isolation stage	8
Maximum number of migrated solutions per species	1
Virtual population	
Lethal age (in generations)	35

6.1 Case Study 1: Weight vs Energy minimization

The results obtained for the first case study demonstrate the expected trade-off between weight (Eqs. 3.58) and energy (Eq. 3.59) minimization objectives. Table 6.4 presents some statistics related to the solutions generated during this optimization process. A reasonable number of solutions on the Pareto front was obtained for this experiment, in line to previous experience [2]. The difference between the number of solutions that were generated and those that were evaluated are due to the newly implemented approach to similarity control, which excludes

a priori those solutions found to be too similar to the ones already present in SP_i without evaluating them.

Table 6.4: Summary of results - 1^{st} case study

statistic	value
n of generated solutions	21081
n solution evaluations	17443
n of solutions in VP	207
n of Pareto-optimal solutions	43

The Pareto front obtained for this case study is presented in Fig. 6.3 and, apart from the two objectives that were considered, also includes the corresponding values for the third objective, defined in Eq. 3.60, as well as the ratio between the energy and its standard deviation, both plotted on the right *y* axes. A high inter-dependence of the second and third objectives can be observed. However, their ratio appears to be less correlated to the energy.

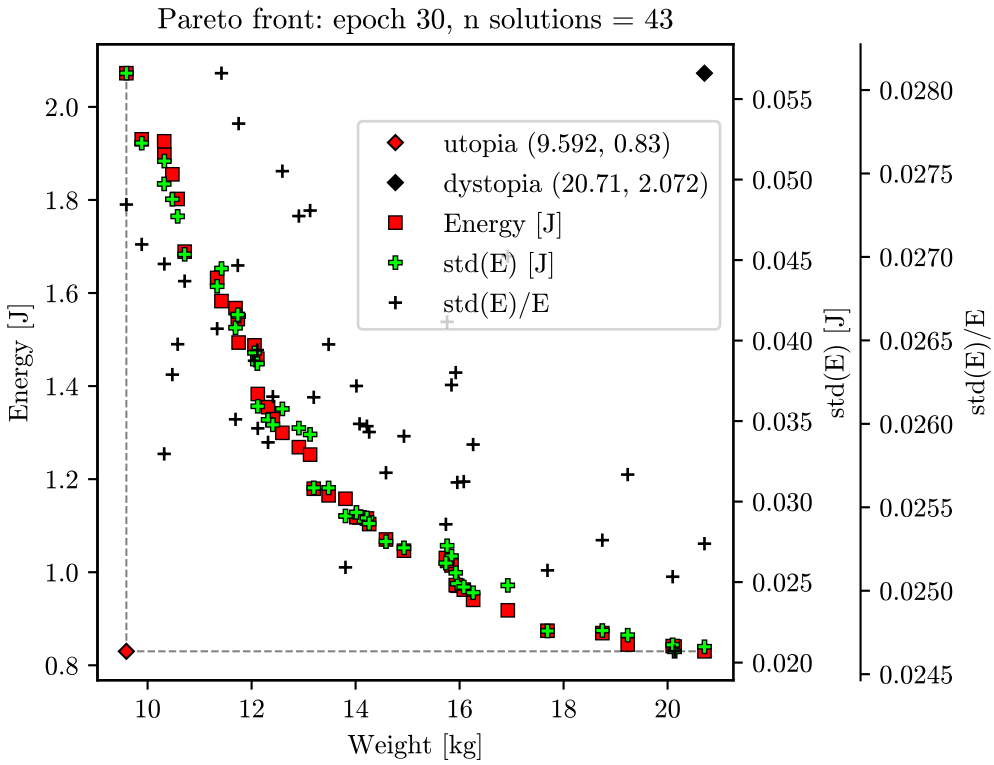


Figure 6.3: Pareto front - 1^{st} case study

Fig. 6.4 shows the history of the successive Pareto fronts obtained throughout the evolutionary process, registered at the end of each epoch. It is noticeable how each successive epoch results in a Pareto front with gradually diminishing improvements. This convergence indicates that the

optimization process potentially reached its limit, which is also illustrated by the convergence of the utopia points near the last epoch.

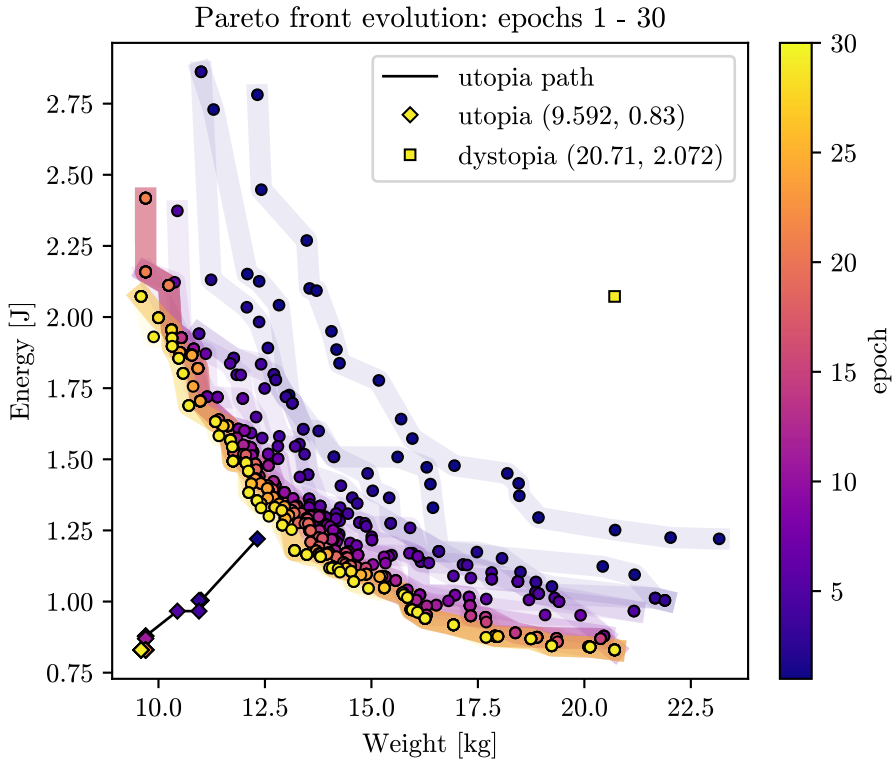


Figure 6.4: Evolution of the Pareto front throughout the epochs 1st case study

6.2 Case Study 2: Weight vs Energy variability minimization

The results obtained for the second case study demonstrate the expected trade-off between the minimization of weight (Eq. 3.58) and energy standard deviation (Eq. 3.60). As expected, given the high correlation, the Pareto fronts obtained for the first and second case studies contain a similar number of solutions, all program inputs being equal (Table 6.5).

Table 6.5: Summary of results - 2nd case study

metric	value
n of generated solutions	20985
n solution evaluations	17431
n of solutions in VP	228
n of Pareto-optimal solutions	48

Fig. 6.5 shows how, as in the first case study, the second objective, plotted on the left y axis, appears to be highly correlated to the third objective, plotted on the leftmost right y axis. To

further illustrate this high correlation, results for the 3OOP are later presented, which should corroborate the findings from the previous runs by outputting a three-dimensional Pareto front which, despite being, by definition, a surface, closely resembles a curve.

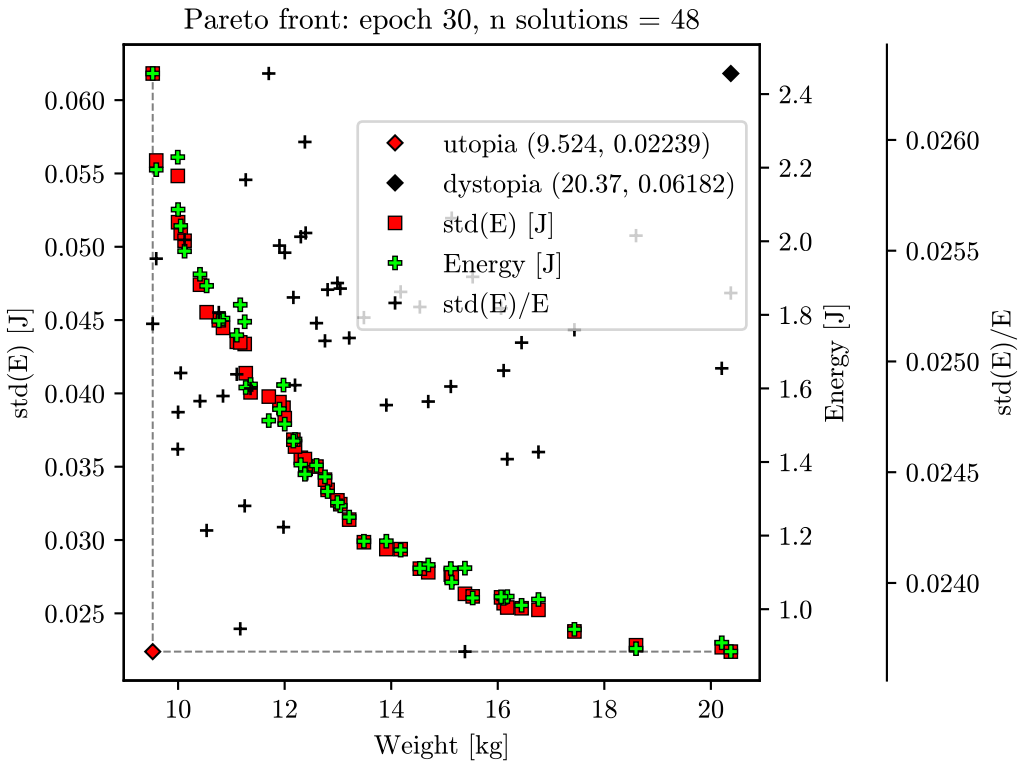


Figure 6.5: Pareto front - 2^{nd} case study

Fig. 6.5 shows similar convergence of the Pareto front to the first case study 6.1.

6.3 Case Study 3: Tri-objective optimization problem

Even though the results for the 3OOP could be interpreted as redundancy of the third case study given the added complexity of defining and including the third objective in the OA and the resulting increased computational expense, it should be noted that, owing to the fact that there is not a direct correspondence between the second and third objectives as visible in the plot of the Pareto front (refer to the plot of the second case study where the 3rd obj is added), the 3OOP leads to additional Pareto-optimal solutions, which would otherwise be near the Pareto front of either the first or second case studies but nevertheless dominated. This is a consequence of these solutions not being given the chance of expressing their valuable trade-offs in a higher-dimensional space.

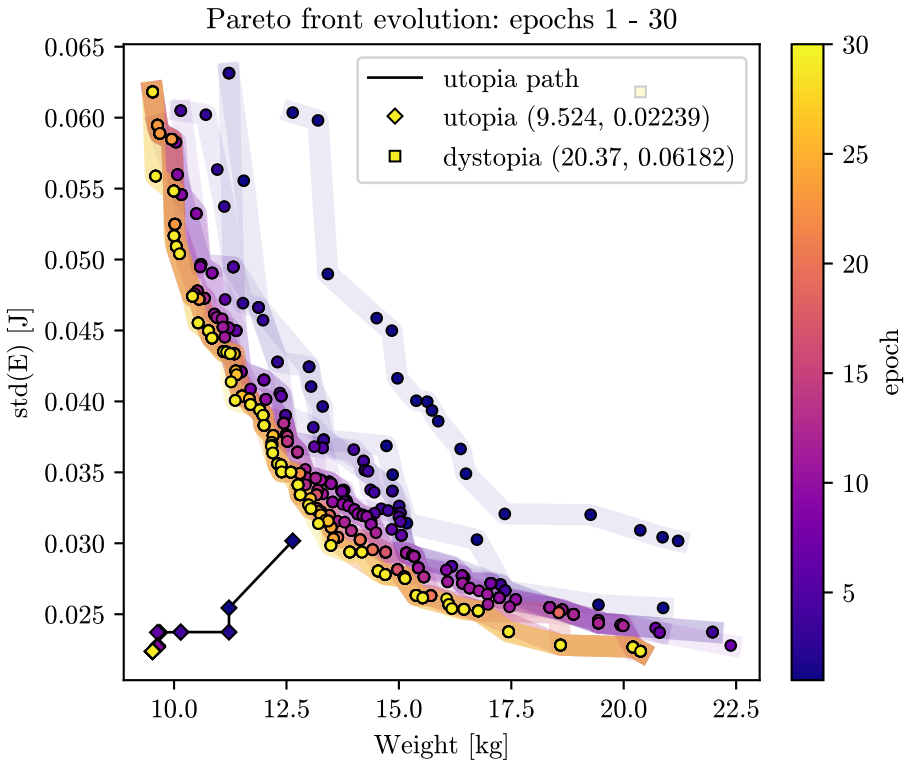


Figure 6.6: Evolution of the Pareto front throughout the epochs - 2^{nd} case study

6.4 Case Study 4: an iteration of the original 3OOP

This new formulation of the 3-objective optimization problem considers the minimization of the ratio between the standard deviation of the strain energy and its mean value as the third objective. As expected, this formulation leads to a greater independence between the second and third objectives, which can be seen in the considerably higher number of Pareto-optimal solutions obtained.

Table 6.6: Summary of results - 4^{th} case study

metric	value
n of generated solutions	20989
n solution evaluations	17455
n of solutions in the VP	982
n of Pareto-optimal solutions	780

The present results were derived after applying some changes to the program's logic, which proved successful. Namely, the criterion for replacing solutions due to similarity in each SP was updated to only discard solutions that were found to be clones of any solution previously generated by the program. This is equivalent to admitting a minimum Hamming distance of 1,

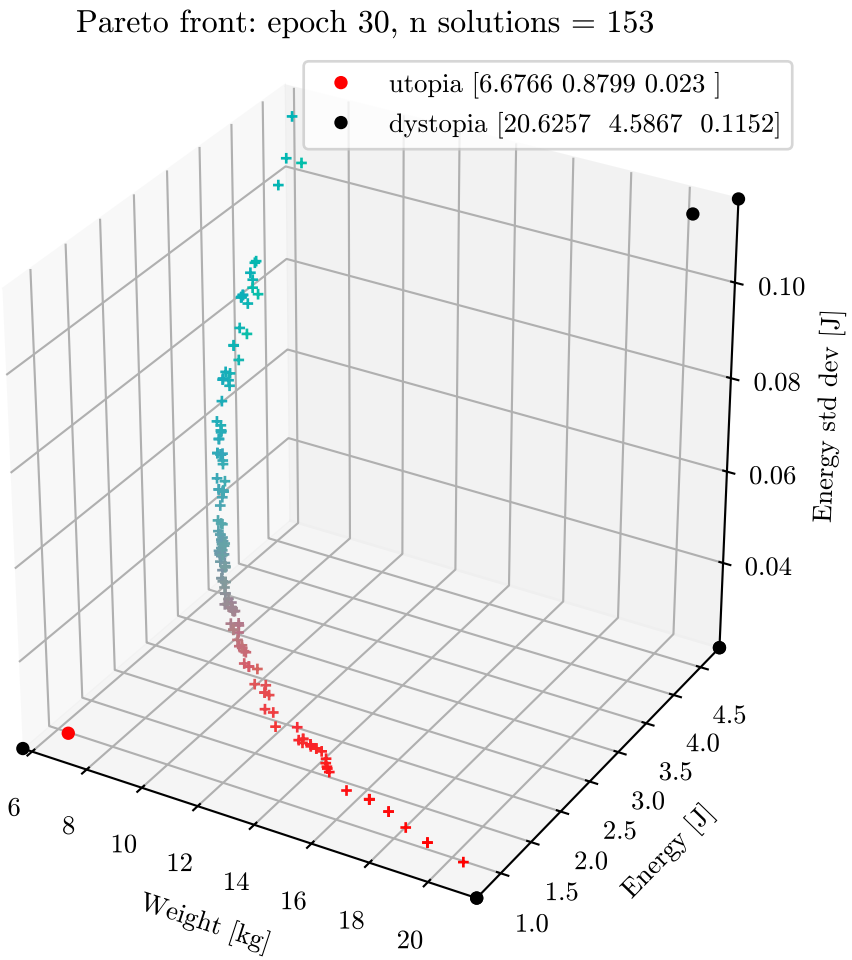


Figure 6.7: Pareto front - 3^{rd} case study

whereas the previous program's definition selected for a minimum Hamming distance of 8. The basis for this update was the notion that asserting a level of genetic dissimilarity greater than what is required to avoid introducing clones inevitably excludes several potentially-optimal solutions in the neighborhood of the existing solutions, since similarity control is applied before any solution is evaluated, to avoid unnecessary computations. Such behavior appears in contradiction to the discretization of the design space that was established for the optimization model (6.3), which itself defines the desired set of solutions to explore. This change did result in a considerably greater number of Pareto-optimal solutions: 780 vs. 464 (Table 6.6).

Fig. 6.8 shows the evolution of the number of Pareto-optimal solutions and size of the VP at the end of each epoch. The stable increase of the number of Pareto-optimal solutions up until the very last epoch indicates that the MOMA might still find further optimal solutions by simply increasing the number of epochs.

Representing the 3-dimensional Pareto front in an intuitive manner presented a challenge

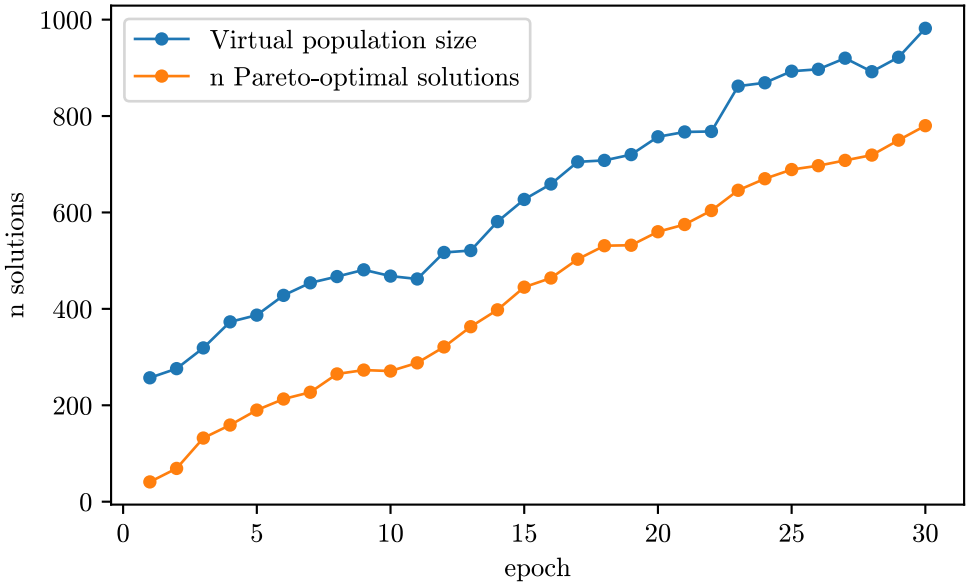


Figure 6.8: Number of Pareto-optimal solutions and size of the VP at the end of each epoch

in itself, given that its discrete nature implies a non-trivial surface interpolation. Such approach was attempted, but deemed inadequate nonetheless, since the spatial distribution of the Pareto-optimal solutions does not closely resemble a regular grid. It was decided to adopt an RGB color representation where each of the three color components is given by the distance from the utopia point along a given axis (x - red, y - green and z - blue), normalized by the difference between the utopia and dystopia points, to add sense of depth. Additionally, a line segment starting at each point indicates the direction to the dystopia point (Fig. 6.9).

Fig. 6.10 displays the same Pareto front as Fig. 6.9 but with different colors representing solutions belonging to different species. In order to ensure enough contrast, the colors were defined by alternating two colormaps ('viridis' and 'plasma'). It is noticeable how solutions belonging to the same species (Definition 4.2.1) tend to group together in different areas of the Pareto front, analogous to how *true* species, in the biological sense, usually present specific adaptations to different environments. This well-defined separation between different species illustrates the suitability of the adopted definition.

Fig. 6.11 shows the position distribution per design variable, as well as their respective modes.

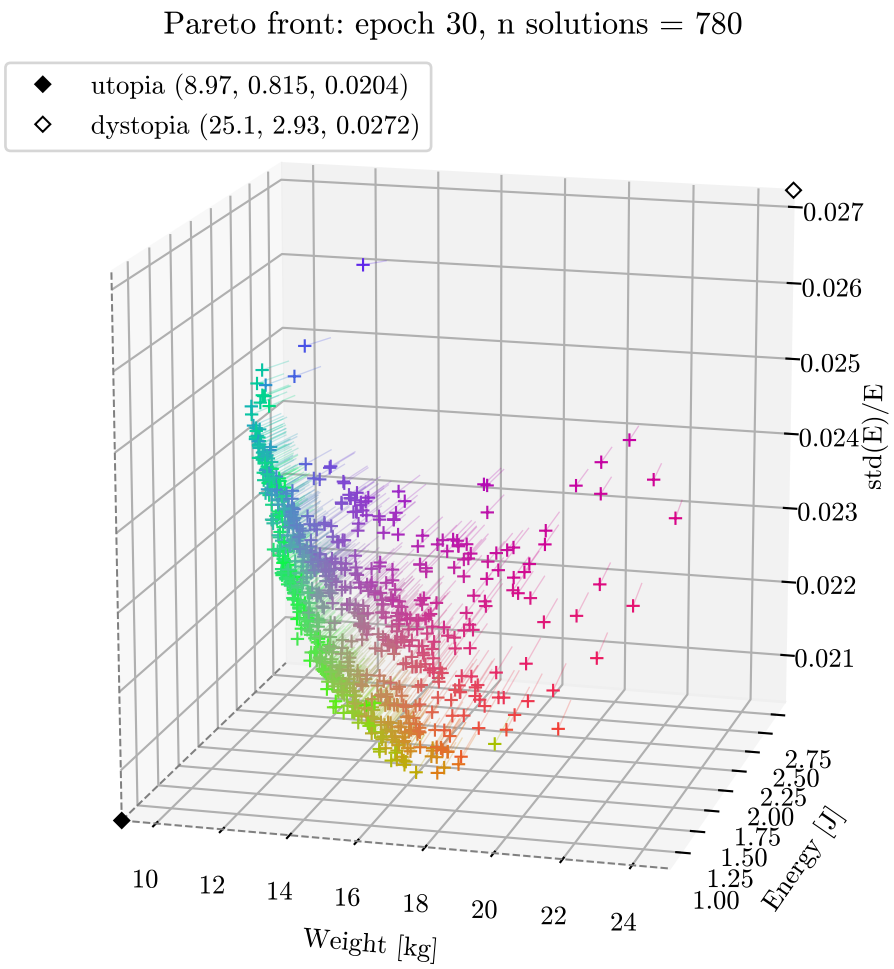


Figure 6.9: Pareto front - 4^{th} case study. The color of each point was defined using RGB code, where each parameter corresponds to the distance from the utopia point along each axis, normalized by the difference between the utopia and dystopia points.

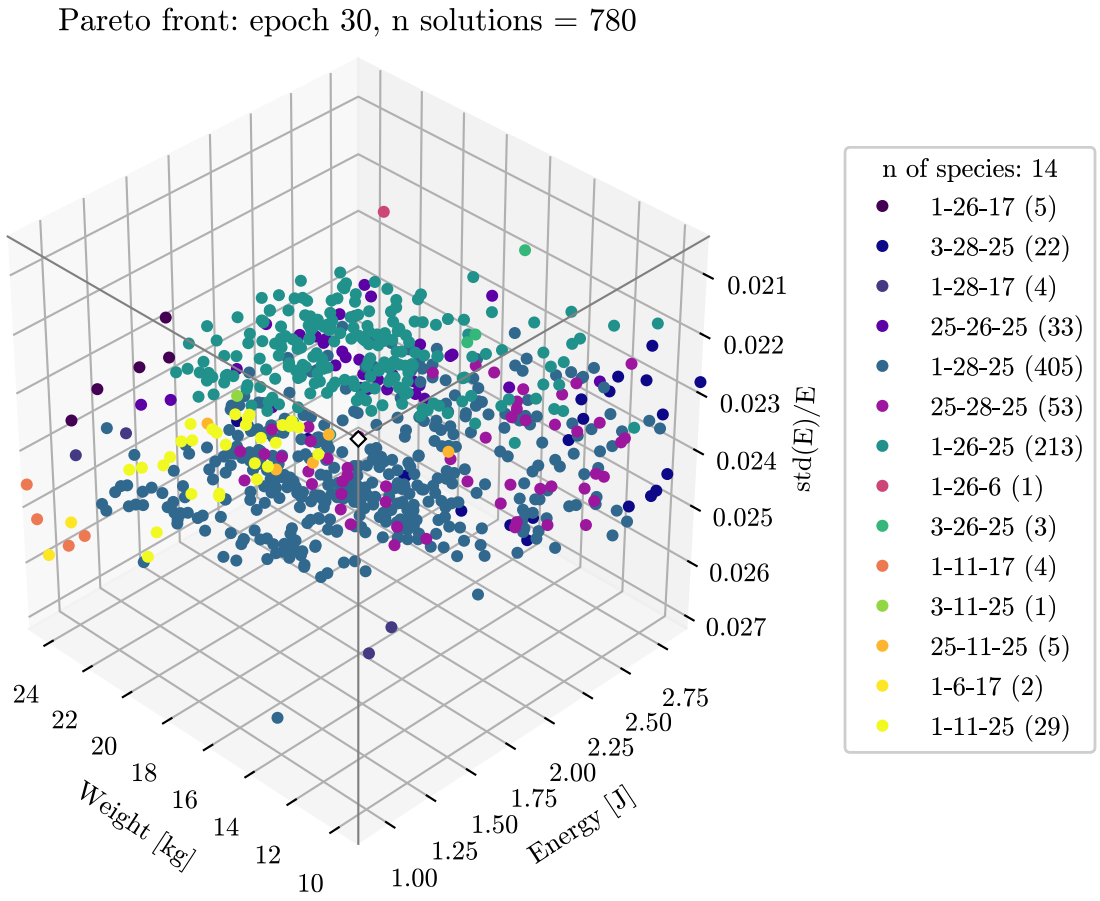


Figure 6.10: Species representation on the Pareto front - 4^{th} case study

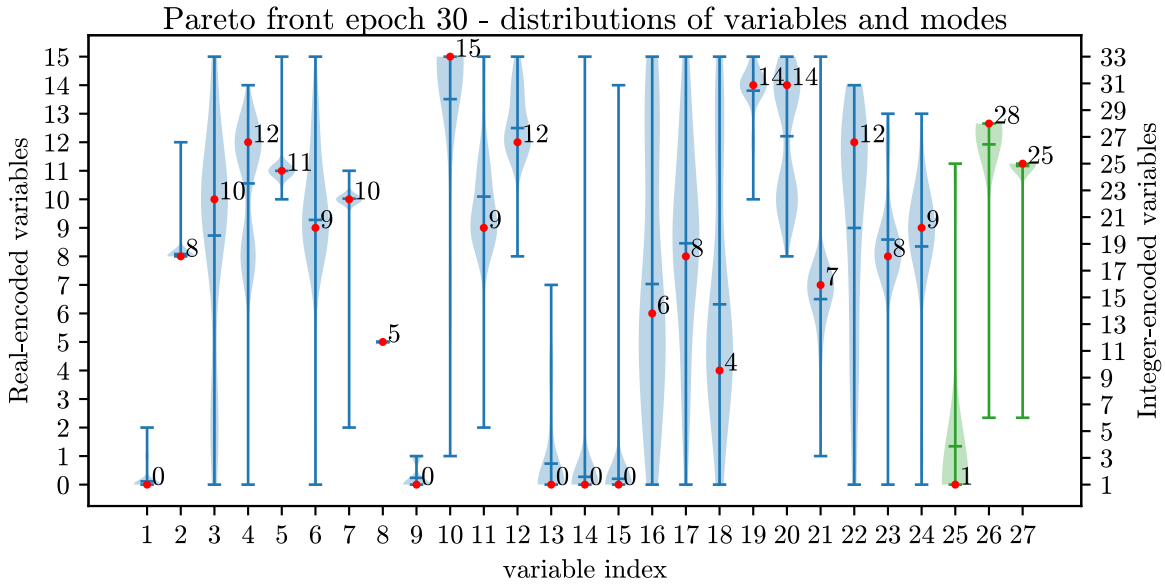


Figure 6.11: Position distribution per variable considering the Pareto-optimal solutions. Red markers indicate the mode of each variable's positions.

Chapter 7

Conclusions

A framework for the multi-objective optimization of composite stiffened structures with nonlinear geometric behavior was developed. The underlying approach involved the application of concepts such as the determination of the equilibrium path through the arc-length control method, the computation of stiffness considering the discretization of the structure into layers made of different composite materials and the resulting anisotropy and the analysis of structural integrity through an approach which unifies buckling and FPF. The MOMA was applied to four MOOPs of a beam-stiffened shell composite structure, and the results that were obtained illustrate its effectiveness in finding optimal solutions.

Finite element program

The finite element program used in the underlying analysis was developed from an existing standalone Fortran program which was developed independently from an optimization context, and therefore presented an opportunity for increasing efficiency since some definitions of the structure being optimized are shared by all resulting variations that correspond to different solutions. One such example is geometry interpolation, more specifically the definition of the linear components of the element strain matrices (and correspondingly their transposes), whose values are now obtained before calling the optimization program and stored in global variables to be used in all computations. As a result, only the nonlinear components of each $\mathbf{B}^e$ need to be updated, since they are displacement-dependent and therefore dictated by each solutions' phenotype (all boundary conditions remaining equal). Additionally, the previously implemented procedure of computing each individual component (both linear and nonlinear) and storing them in `unformatted` files was replaced with local variables.

The aforementioned changes raised the challenge of dealing with a considerable amount of

matrix definitions pertaining to each structure definition without resorting to both files and global variables, given that the absence of *locality* and the resulting side effects prevent the compiler from producing a more efficient application [51]. Furthermore, in a high-performance computing (HPC) context, especially suited for finite element applications due to their heavy reliance on often inter-independent intensive matrix computations, parallelization of element stiffness matrix evaluation functions can only be achieved if they obey these functional programming rules. Accordingly, the introduction of derived types enabled these entities to be passed-by-value as a single data structure as opposed to their separate handling. The original frontal solver was also replaced with the Gauss elimination method, since the former relied on `read`/`write` operations which compromise efficiency, and the new method was validated using the LAPACK library [48].

Optimization algorithm

The MOMA was extended to be able to handle optimization problems with any number of objectives. Subsequently, the 3OOP addressing all three objectives that were initially defined was studied. The identification of a high degree of correlation between the mean value of energy (E) and its standard deviation (std(E)) led to a new objective being defined, which corresponds to the relative value of the standard deviation of energy (std(E)/E). Apart from resulting in a considerably higher number of Pareto-optimal solutions, this formulation is supported by the notion that, since std(E) is used to quantify the robustness of the structure with respect to uncertainties in the manufacturing process, its value should be normalized.

The study of species distribution on the Pareto front revealed a niching effect which validated the adopted domain decomposition that was based on the assumption that different material combinations offer specific trade-offs. Furthermore, this effect suggests that hybrid composites may unlock otherwise unattainable design solutions.

An interface for the automatic generation of graphs from the results of the OA was developed in Python.

7.1 Future works

Given the highly dynamic process of developing an OA, the MOMA still has considerable potential for improvement. Technically, this may be considered a non-ending process, since relying on heuristics means constantly trying to extract and decipher hidden relationships from one or more sets of problems.

HPC prospects

The developments regarding the finite element program set the stage for implementing a version of the program that resorts to GPU offload in order to leverage the highly parallel nature of modern GPUs, through the use of the OpenMP® API integrated into the Intel® Fortran Compiler [51].

Meme propagation

Currently, meme attribution is based on past performance of each operator; alternatively, meme inheritance could be added. The previous mechanism based on past success rate may be kept and the crossover group could simply be divided into two groups, one for each type of meme attribution. In such scenario, the performance of the two mechanisms could be compared, and another rule could be implemented which would adjust the relative sizes of these two groups based on the performance of their mechanisms. The evolution history feature that was added could be used to study the genealogy or family tree of the solutions, in order to try to identify tendencies in meme propagation and consequently the levels of synergy between operators, which could also be used in the assignment of offspring memes.

Convergence criteria

Another opportunity would consist of implementing new convergence metrics, such as measuring the percentage of new Pareto-optimal solutions brought by each additional epoch, or the decrease of the success rate of the operators, in order to appropriately define the length of execution.

New crossover operators

A new crossover that necessarily selects two parents belonging to the same species is proposed, under the assumption that same-species parents are more likely to be genetically similar and therefore generate fitter offspring. This proposal is based on the assumption that each species constitutes a local optimum that corresponds to a very narrow interval of the design space associated with real-type variables. Similarly, severe discontinuities must exist between different species. Crossover between too distant individuals can lead to a decrease in fitness of the offspring, known as *outbreeding depression*. In contrast, selecting for similar genetic traits constitutes an example of positive assortative mating, which can improve offspring fitness in some cases but also result in decreased fitness when the parents are too similar - *inbreeding depression*.

A new crossover operator is suggested that would select two parents (p_1 and p_2) which are consecutive on the Pareto front. This would mean that p_1 would be randomly selected from all Pareto-optimal solutions, and p_2 would be sorted from the solutions adjacent to p_1. This proposal is based on the heuristic that solutions that are close in the Pareto front tend to display greater levels of similarity (and therefore could lead to fitter offspring), and could be interpreted as an 'automatic' decomposition of the design space imposed on the solution's evaluations, rather than their phenotypes.

Bibliography

[1] Carlos Conceição António. A multilevel genetic algorithm for optimization of geometrically nonlinear stiffened composite structures. *Structural and Multidisciplinary Optimization*, 24:372–386, 2002.

[2] Carlos Conceição António. Memeplex-based memetic algorithm for the multi-objective optimal design of composite structures. *Composite Structures*, 329:117789, 2024.

[3] Carlos Conceição António. *Otimização de Sistemas em Engenharia - Fundamentos e algoritmos para o projeto ótimo*. 2020.

[4] John H Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, 9(3):297–314, 1962.

[5] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826(1989):37, 1989.

[6] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[7] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[8] Ryan Meuth, Meng-Hiot Lim, Yew-Soon Ong, and Donald C Wunsch. A proposition on memes and meta-memes in computing for higher-order learning. *Memetic Computing*, 1:85–100, 2009.

[9] William E Hart, Natalio Krasnogor, and James E Smith. Recent advances in memetic algorithms. 2006.

[10] Richard Dawkins. *The selfish gene*. Oxford university press, 2016.

[11] Natalio Krasnogor and Jim Smith. A memetic algorithm with self-adaptive local search: Tsp as a case study. In *Proceedings of the 2nd annual conference on genetic and evolutionary computation*, pages 987–994, 2000.

[12] Minh Nghia Le, Yew-Soon Ong, Yaochu Jin, and Bernhard Sendhoff. Lamarckian memetic algorithms: local optimum and connectivity structure analysis. *Memetic Computing*, 1:175–190, 2009.

[13] Jean-Baptiste de Monet de Lamarck. *Philosophie zoologique: ou exposition des considérations relatives à l'histoire naturelle des animaux...*, volume 2. Dentu, 1809.

[14] James Mark Baldwin. A new factor in evolution. *Diacronia*, (7):1–13, 2018.

[15] Peter D Turney. Myths and legends of the baldwin effect. *arXiv preprint cs/0212036*, 2002.

[16] Reiji Suzuki and Takaya Arita. The dynamic changes in roles of learning through the baldwin effect. *Artificial Life*, 13(1):31–43, 2007.

[17] Natalio Krasnogor and James Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[18] Amir-massoud Farahmand, Majid Nili Ahmadabadi, Caro Lucas, and Babak N Araabi. Interaction of culture-based learning and cooperative co-evolution and its application to automatic behavior-based system design. *IEEE transactions on evolutionary computation*, 14(1):23–57, 2009.

[19] Quang Huy Nguyen, Yew-Soon Ong, and Natalio Krasnogor. A study on the design issues of memetic algorithm. In *2007 IEEE Congress on Evolutionary Computation*, pages 2390–2397. IEEE, 2007.

[20] Quang Huy Nguyen, Yew Soon Ong, and Meng Hiot Lim. Non-genetic transmission of memes by diffusion. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1017–1024, 2008.

[21] Natalio Krasnogor. *Studies on the theory and design space of memetic algorithms*. PhD thesis, University of the West of England at Bristol, 2002.

[22] Graham Kendall, Peter Cowling, Eric Soubeiga, et al. Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, pages 667–671. Citeseer, 2002.

[23] Yew Soon Ong and Andy J Keane. Meta-lamarckian learning in memetic algorithms. *IEEE transactions on evolutionary computation*, 8(2):99–110, 2004.

[24] Yew-Soon Ong, Meng-Hiot Lim, Ning Zhu, and Kok-Wai Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(1):141–152, 2006.

[25] Natalio Krasnogor and Steven Gustafson. A study on the use of"self-generation"in memetic algorithms. *Natural Computing*, 3:53–76, 2004.

[26] Jim E Smith. Coevolving memetic algorithms: A review and progress report. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):6–17, 2007.

[27] Jacob L. Pelletier and Senthil S. Vel. Multi-objective optimization of fiber reinforced composite laminates for strength, stiffness and minimal mass. *Computers & Structures*, 84(29):2065–2080, 2006.

[28] Carlos Conceição António. A hierarchical genetic algorithm with age structure for multimodal optimal design of hybrid composites. *Structural and Multidisciplinary Optimization*, 31:280–294, 2006.

[29] Rahul, G. Sandeep, D. Chakraborty, and A. Dutta. Multi-objective optimization of hybrid laminates subjected to transverse impact. *Composite Structures*, 73(3):360–369, 2006.

[30] H Baier, M Huber, and H Langer. Design optimization of hybrid material structures. *Structural and Multidisciplinary Optimization*, 36:203–213, 2008.

[31] H Ghiasi, D Pasini, and L Lessard. Pareto frontier for simultaneous structural and manufacturing optimization of a composite part. *Structural and Multidisciplinary Optimization*, 40:497–511, 2010.

[32] F.-X. Irisarri, F. Laurin, F.-H. Leroy, and J.-F. Maire. Computational strategy for multiobjective optimization of composite stiffened panels. *Composite Structures*, 93(3):1158–1167, 2011.

[33] Carlos Conceição António. Local and global pareto dominance applied to optimal design and material selection of composite structures. *Structural and Multidisciplinary Optimization*, 48:73–94, 2013.

[34] Carlos Conceição António. A memetic algorithm based on multiple learning procedures for global optimal design of composite structures. *Memetic Computing*, 6:113–131, 2014.

[35] A Pica, RD Wood, and E Hinton. Finite element analysis of geometrically nonlinear plate behaviour using a mindlin formulation. *Computers & Structures*, 11(3):203–215, 1980.

[36] José Augusto Trigo Barbosa. *Análise não linear por elementos finitos de placas e cascas reforçadas: aplicação a laminados anisotrópicos*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 1992.

[37] Carlos Conceição António. Optimisation of geometrically non-linear composite structures based on load–displacement control. *Composite Structures*, 46(4):345–356, 1999.

[38] MA1913032 Crisfield. An arc-length method including line searches and accelerations. *International journal for numerical methods in engineering*, 19(9):1269–1289, 1983.

[39] Bernard Budiansky. Theory of buckling and post-buckling behavior of elastic structures. *Advances in applied mechanics*, 14:1–65, 1974.

[40] Stephen W Tsai. *Composites design*, volume 5. Think composites Dayton, OH, 1988.

[41] Carlos Conceição António and IA Lhate. A hybrid crossover operator for structural optimisation based on commonality in genetic search. *Engineering Computations*, 20(4):390–408, 2003.

[42] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[43] Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjective optimization: formulationdiscussion and generalization. In *Icga*, volume 93, pages 416–423. Citeseer, 1993.

[44] Carlos Conceição António. A hierarchical genetic algorithm for reliability based design of geometrically non-linear composite structures. *Composite Structures*, 54(1):37–47, 2001.

[45] Carlos Conceição António, A Torres Marques, and JF Gonçalves. Reliability based design with a degradation model of laminated composite structures. *Structural optimization*, 12:16–28, 1996.

[46] S.S. Rao. *Reliability-based Design*. McGraw-Hill, 1992.

[47] Bruce M. Irons. A frontal solution program for finite element analysis. *International Journal for Numerical Methods in Engineering*, 2(1):5–32, 1970.

[48] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[49] Developer Reference for Intel® oneAPI Math Kernel Library for Fortran. https://www.intel.com/content/www/us/en/docs/onemkl/developer-reference-fortran/2024-1/overview.html. Accessed: 2024-06-20.

[50] Automatic Parallelization with Intel® Compilers. https://www.intel.com/content/www/us/en/developer/articles/technical/automatic-parallelization-with-intel-compilers.html. Accessed: 2024-06-20.

[51] Intel® Fortran Compiler Classic and Intel® Fortran Compiler Developer Guide and Reference. https://www.intel.com/content/www/us/en/docs/fortran-compiler/developer-guide-reference/2024-1/overview.html. Accessed: 2024-06-20.