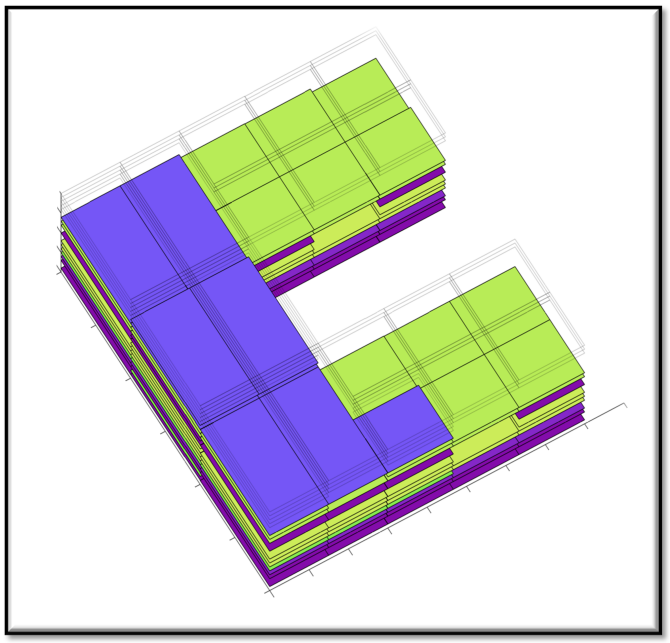
Opti-BLESS - An Open-source Toolbox for the Optimisation of Blended Stacking Sequences (MATLAB)

A User Manual

Release V 1.0.0

T. Macquart



Abstract

Although the use of composites in primary structures has grown significantly over the past decades, the optimisation of composite structures remains quite challenging. In this regard, many companies and academics have developed their own in-house software for composite optimisation. Whilst these in-houses applications have contributed to highlighting the benefits of composite tailoring, they remain closed proprietary codes. By contrast, the toolbox presented is this report provides a first step towards the development of an open-source free tool for optimising composite structures. The **Opti**misation of **BLE**nded **S**tacking **S**equences Toolbox (**Opti-BLESS**) is a small and easy-to-use code for the optimisation of composite structures. **Opti-BLESS** wraps a collection of functions for laminate designs with the MATLAB optimisation toolbox in order to achieve a compact user interface for composite structure optimisation.

This report serves as a users manual for **Opti-BLESS** and includes capability overviews and procedures for software execution, as well as a tutorials.

Preface

The **Opti-BLESS** idea started in 2015 as part of my internal research work at Delft University of Technology. The original goal was to develop a short code capable of retrieving stacking sequences from lamination parameter or stiffness based designs. I soon realised that, although different methods had been proposed for composite optimisation, these approaches were not easily accessible. I, therefore, needed to write my own code which I later on decided to polish and make freely available for others no to have to go through the same process. This document serves as a user introduction to the **Opti**misation of **BLE**nded **S**tacking **S**equences Toolbox (**Opti-BLESS**) developed for MATLAB. The toolbox development is mostly a free-time side project for me nowadays and much interesting features and idea remains to be tested. In particular, the development a composite optimisation tool that would significantly reduce the discrepancies between optimised and manufacturable designs comes to mind. Of course everyone is welcome to simply use and/or contribute to the toolbox development. Feedback is also always appreciated.

Requirement
MATLAB 2014.b or more recent
MATLAB optimization_toolbox
MATLAB gads_toolbox
MATLAB signal_toolbox

Contact Information:
TMacquart.ssort@gmail.com
Opti-BLESS Toolbox (Only private users for now. Add pulbic link later)

Nomenclature

BOLD	=	Bold variables indicate vector
$\boldsymbol{N}$	=	Number of plies per laminate
$\boldsymbol{\theta}$	=	Ply angles
$\boldsymbol{L_{(-\theta)}}$	=	Balanced ply angle pairs locations in the laminate
$\boldsymbol{L_{(0/\pm 45/90)}}$	=	10% rule ply angle locations in the laminate
$\boldsymbol{\theta_M}$	=	Mid-plane laminate ply angles
$\boldsymbol{\Xi}$	=	Ply insertion location in the laminate
$\boldsymbol{\alpha}$	=	Scaling coefficients
$[A]$	=	Laminate in-plane stiffness matrix
$[B]$	=	Laminate coupled stiffness matrix
$[D]$	=	Laminate out-of-plane stiffness matrix
h	=	Laminate thickness
$[\bar{D}]$	=	Transformed ply stiffness matrix
t_{ply}	=	Ply thickness
N_{lam}	=	Number of laminates
SS	=	Stacking Sequence
$\boldsymbol{LP}$	=	Lamination parameters
$V_1^A, V_2^A, V_3^A, V_4^A$	=	In-plane lamination parameters
$V_1^B, V_2^B, V_3^B, V_4^B$	=	Coupled lamination parameters
$V_1^D, V_2^D, V_3^D, V_4^D$	=	Out-of-plane lamination parameters

Short File Descriptions

Attribute_NDvs.m	=	Attribute the number of design variables to each genotype field
Check_Feasibility	=	Check the individual feasibility w.r.t. non explicit design guidelines
ComputeSSTable.m	=	Constructs the stacking sequence table based on decoded genotype
Convert_Genotype.m	=	Convert Genotype into SSTable using ComputeSSTable
Convert_SS2LP.m	=	Convert stacking sequence into lamination parameters
Convert_SS2ABD.m	=	Convert stacking sequence into its corresponding A,B,D stiffness matrices
Eval_Fitness.m	=	Wrapping function that receive the coded Individual and evaluate its fitness
FormatInput.m	=	Read and format inputs structures
ga.m	=	Matlab GA function
Generate_IniPop.m	=	Generate the initial population - Satisfying design guidelines
OptiBLESS.m	=	Main toolbox function. Deals with inputs, run optimisation and return optimised results.

Chapter 1

Introduction

1 Motivation

The significant weight saving potential achievable with tailored composite structures is now well-recognised amidst the scientific community. The incentive to manufacture strong yet lightweight structures is also resulting in the increasing use of composite materials in many engineering applications. Moving from metals to composite structures has, however, brought forward a considerable new set of challenges including new failure mechanisms, added complexity and increased number of design variables.

The optimisation of composite structure is challenging. One can roughly distinguish between three optimisation frameworks:

1 - Single Step Direct Optimisation
2 - Multi-step Direct Optimisation
3 - Multi-Step Indirect Optimisation

Direct optimisation refers to the use of explicit composite properties such as ply angles and the number of plies. Using explicit properties as design variables allows manufacturing constraints to easily be integrated in the direct optimisation framework [1]. This is the direct framework strong selling point and probably the reason why most industries still use this approach. On the other hand, direct optimisation is challenging because of the design space high-dimensionality, non-convexity and discrete nature. As a results, the type of optimisers that can be used are limited and a thorough exploration of the design space, even for medium scale problems, is computational intractable. Reducing the number of design variables would result in a reduction of composite tailoring and would therefore be counter-effective to the improvement of composite designs.

In comparison to the direct framework, one can also find indirect composite optimisation approaches. In this case, indirect refers to the use of an intermediate design parametrisation. Typical parametrisations include stiffness matrices, lamination parameters and polar invariants. The key advantage of indirect approaches, excluding polar invariants, is the reformulation of the composite optimisation problem into a 'nicer' continuous space for which robust and efficient optimisers can handle large scale problems at reasonable computational costs [2]. As a result, the design space can be thoroughly explored and exploited to maximise composite tailoring. The challenging part, however, is to retrieve manufacturable laminates from the optimised parametrised designs. In particular, the current incapability of parametrised design spaces to integrate manufacturing constraints results in solutions that are not guaranteed to have equivalent in the explicit design space.

An ideal composite optimisation framework should combine the advantages of both approaches and be able to optimise large composite structures while considering manufacturing constraints at reasonable computational costs. It is likely that the direct optimisation framework, due to its nature, will never be able to effectively handle large scale composite optimisation and that the use of indirect approaches will therefore become prevalent. Additionally, efforts have already been made in order to integrate some manufacturing constraints into parametrised spaces [4, 3]. However, the difficulties arising in retrieving laminates from the parametrised spaces are still problematic and have originally motivated the development of the **Opti**misation of **BLE**nded **S**tacking **S**equences (**Opti-BLESS**) Toolbox. Nevertheless, it likely that both the direct and indirect frameworks will be used in the future and as a result

Opti-BLESS provides both optimisation capabilities. The aim of the developed toolbox is to provide an easy-to-use tool for the optimisation of blended stacking sequences.

2 Toolbox Capabilities

Opti-BLESS has the following capabilities:

- Retrieves stacking sequences from a set of stiffness matrices (indirect framework)
- Retrieves stacking sequences from a set of lamination parameters (indirect framework)
- Optimises stacking sequences (direct framework)

In order to find the best stacking sequences, **Opti-BLESS** allows the **laminates' thicknesses**, **fibre angles** and **ply insertion location** to be optimised using the genetic algorithm provided by MATLAB. Additionally, the following list of design guidelines [5] for composites can be used:

List of Design Guidelines

1. **Symmetry**. Stacking sequence is mirrored about the mid-plane.
2. **Balance**. All fibre angles, except 0° and 90°, occur in ± pairs.
3. **Damtol**. Damage Tolerance, ±45° plies are used for the upper and lower laminate plies.
4. **Rule10percent**. A minimum of 10% of plies in each of the 0°, 45° and 90° is enforced.
5. **Disorientation**. The change of angles between two consecutive plies should not exceed 45°.
6. **Contiguity**. No more that 'X' plies with same fibre orientation should be next to each other (set by user).
7. **DiscreteAngle**. Discrete fibre angles are used (possible values are set with $\Delta Angle$).
8. **InernalContinuity**. One ply must be kept spanning the entire structure every 'X' plies (set by user).
9. **Covering**. Covering plies on the lower and upper surfaces of the laminate cannot be dropped.

3 Toolbox Structure

Figure 1.1 shows an overview of the algorithm of **Opti-BLESS**. As it can be seen, users have very little data to process because most of the code serves as a wrapper around the MATLAB genetic algorithm.

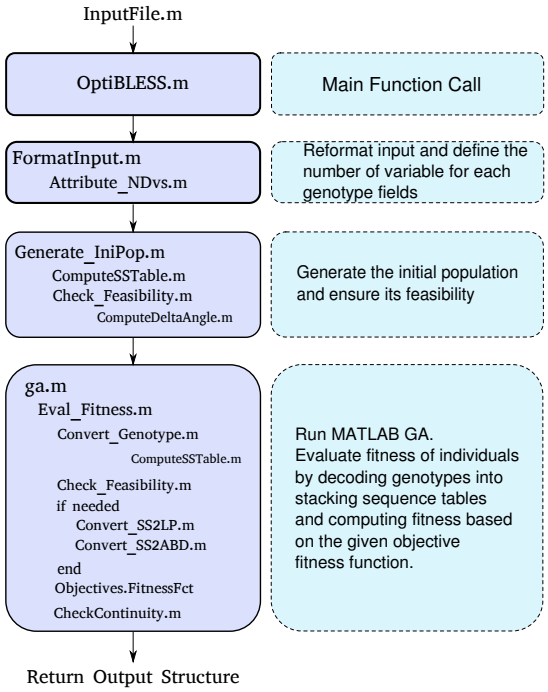


Figure 1.1: **Opti-BLESS** algorithm overview

The input file contains three structures, namely objective, constraints and options. Once the three structures are defined, the main function (i.e. **OptiBLESS**) is called. This process is fully automated and will return an output structure containing the best found set of stacking sequences. The objective structure mostly includes information required for the calculation of the fitness function. Three types of objective structures linked to the different optimisation strategies are available as shown in Figure 1.2:

List of Possible Objectives

Type 1 - Lamination parameters to be matched are given as input and a fitness function based on lamination parameter matching (e.g. root mean square error) is used to evaluate the fitness of stacking sequences evaluated by the genetic algorithm.

Type 2 - Stiffness matrices to be matched $[A]$ $[B]$ and $[D]$ are given as input and a fitness function based on stiffness matrices matching is used to evaluate the fitness of stacking sequences evaluated by the genetic algorithm.

Type 3 - Users can defined their own fitness functions. These can be based on either lamination parameters, stiffness matrices or stacking sequences.

Options for the genetic algorithm optimisation are prescribed into the GA options structure. The following optimisation options are available:

List of Genetic Algorithm Options

1. Population size
2. Number of generations
3. Probability of crossover
4. Percentage of elitism

Other options such as the probability of mutation and the selection function are not directly available as the GA implementation of MATLAB restricts these options for problems with integers. Once all put together the input file include the three input structures as shown in Figure 1.2.

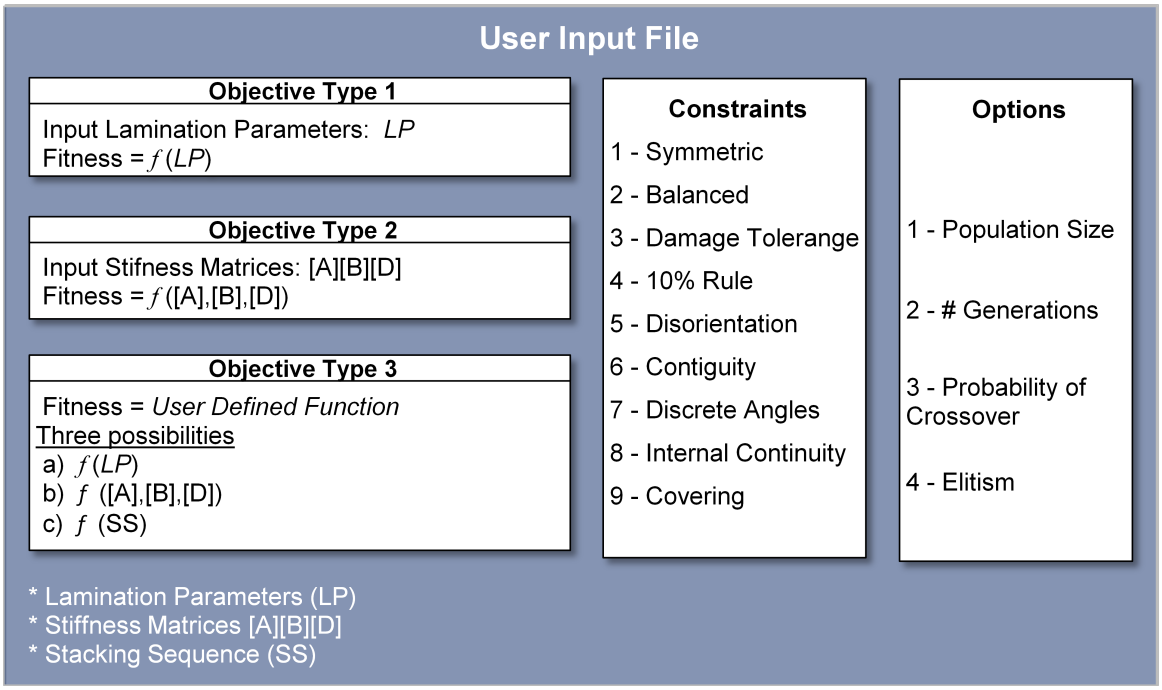


Figure 1.2: **Opti-BLESS** input file overview

4 Notations

4.1 Genotype - Coding of Solutions

As a user you will probably not need to decode the solution yourself, however knowing how individuals are coded in the GA will give you some understanding of the size and limitations related to the design search space. The coding methodology used in **Opti-BLESS** has been developed by the author and inspired from the guide-based approach to composite optimisation [5, 1]. Each individual is coded based on 6 distinct fields of design variables:

1. Number of plies for each patches (**N**)
2. Ply angles ($\boldsymbol{\theta}$)
3. The location of the guide balanced pair angles in the laminate ($\boldsymbol{L_{(-\theta)}}$)
4. The location of angles allocated for the 10% rules (i.e. $\pm$45/0/90) ($\boldsymbol{L_{(-45)}}$,$\boldsymbol{L_{(0)}}$,$\boldsymbol{L_{(90)}}$,$\boldsymbol{L_{(+45)}}$)
5. Ply angles of the mid-plane plies (either 1 or 2 plies) ($\boldsymbol{\theta_M}$).
6. The ply insertion location in the laminate ($\boldsymbol{\Xi}$)

A generic encoded solution is represented as:

$$Genotype = \Big[[N_1 \dots N_{Nlam}][\theta_1 \dots \theta_N][L_{(-\theta_1)} \dots L_{(-\theta_N)}][L_{(0)}\, L_{(+45)}\, L_{(+90^\circ)}\, L_{(-45)} \dots][\theta_{M1}\, \theta_{M2}][\Xi_1 \dots \Xi_D]\Big]$$

Inside the toolbox, genotypes are decoded into their corresponding stacking sequence tables. A stacking sequence table is simply a collection of stacking sequence relate to each other through ply drops or ply insertion. For instance, a generic example of table could be

Table 1.1: Example of stacking sequence table. Ply drops are represented by X's

Number of Plies	**Stacking Sequence**
$N = 5$	$30^\circ/10^\circ/-30^\circ/45^\circ/60^\circ$
$N = 4$	$30^\circ/X/-30^\circ/45^\circ/60^\circ$
$N = 3$	$30^\circ/X/-30^\circ/X/60^\circ$
$N = 2$	$30^\circ/X/X/X/60^\circ$

In **Opti-BLESS** the stacking sequence is computed first. The number of plies of a patch is only used to link the patch with the corresponding row of the stacking sequence table. A few clarifying comments can be made regarding the coding choice:

- $\boldsymbol{\theta}$ - Theta's are the generic fibre ply angles
- Opposite pairs of Theta's angles (i.e balanced angles) are handled explicitly. If the balanced design guidelines is activated the resulting stacking sequence will inherently be balanced. With this approach it is impossible to obtain non-balanced stacking sequences when the balanced guideline is active. The position of balanced angles within the laminate are stored into $\boldsymbol{L_{(-\theta)}}$.
- Similarly to the balanced guideline, the 10% rules design guidelines is handled explicitly. However, both implicit and explicit approach are used in order to ensure the 10% rule is enforced. The locations of angles allocated for this rule are similarly stored in the genotype.
- The mid-plane plies are only allocated if the maximum number of plies define in the objective structure is an odd number. Moreover, this angle will be restricted to 0 or 90 if the balanced design guideline is also active.
- A ply insertion approach has been adopted instead of a ply drop approach. By inserting ply starting from the thinnest laminate it is easier to generate feasible initial populations.
- The upper and lower value of the number of plies per patch have a direct impact on the genotype size. The lowest value is used to construct the thinnest laminate patch while the highest value represent the guide laminate. The stacking sequence is constructed based on these two extreme values.

Based on user options, not all these fields will be used to store design variables. For instance, the number of plies can be fixed or the laminate does not have to be balanced. The following Tables illustrate some of the simplified coding rules that are used in **Opti-BLESS** when the 10% rule design guideline is not activated.

Table 1.2: Numerical representation of typical solutions in **Opti-BLESS**, see Table 1.3 for the corresponding Stacking Sequences ('-' indicate that it is not a design variable)

	Number of Plies	**Theta's**	**Fibre Angle Pair Locations** (only if Balanced)	**Ply Insertion Location**
Example	N_1, N_2, N_3, N_4, N_5	θ_1, θ_2, θ_3, θ_4, θ_5	L_1, L_2, L_3, L_4, L_5	Ξ_1, Ξ_2, Ξ_3, Ξ_4
(A)	-	45°, 90°, 0°, −45°, 45°	-	-
(B)	-	45°, −45°, 45°, 90°,0°,	-	2, 3
(C)	5, 3	45°, −45°, 45°, 90°, 0°	-	2, 3
(D)	-	30°, 45°, 20°, −60°, 10°	2, 8, 5, 4, 6	-
(E)	-	30°, 45°, 20°, −60°, 10°	2, 8, 5, 4, 6	2

Table 1.3: Numerical representation of typical solutions in **Opti-BLESS** (Corresponding Stacking Sequences)

Examples	**Stacking Sequences**
(A) 5-ply Single Lam. Fixed Thickness	Guide = 45°/90°/0°/−45°/45°
(B) 2-Patch Lam. (5 and 3 plies) Fixed Thickness	Guide = 45°/90°/0°/−45°/45° Lam_1 = 45°/-/-/−45°/45°
(C) 2-Patch Lam. (5 and 3 plies) Variable Thickness	same as (B)
(D) 1-Patch Balanced Lam. (10 plies) Fixed Thickness	Guide = 30°/−30°/45°/ 60°/−20°/−10°/20°/−45°/−60°/10°
(E) 2-Patch Balanced Lam. (10 and 8 plies) Fixed Thickness	Guide = 30°/10°/−30°/45°/ 60°/−10°/−20°/20°/−60°/−45° Lam_1 = 30°/ - /−30°/45°/ 60°/ - /−20°/20°/−60°/−45°

The strategy used for inserting ply is not complicated but may seem somewhat confusing at first. Three of the design variable field used insertion, the balanced angles, the 10 % rule angles and new ply insertion. The detailed decoding of a simple individual is now clarified for ply insertion. Take for instance the following genotype composed of 2 patches of 5 and 3 plies:

$$Genotype = \Big[[N_1 = 5\, N_2 = 3][\theta_1\, \theta_2\, \theta_3\, \theta_4\, \theta_5][\Xi_1 = 2\, \Xi_2 = 5]\Big]$$

The thinnest laminate is composed of three plies as given by the second N_2 design variable. The thinnest laminate stacking sequence is simply described by the first three theta's $[\theta_1\, \theta_2\, \theta_3]$. The five ply guide laminate is obtained by inserting the last two theta's $[\theta_4\, \theta_5]$ into the thinnest laminate according the position given by $[\Xi_1\, \Xi_2]$. The insertion is a sequential process, θ_4 is inserted first in position $\Xi_1 = 2$ leading to the intermediate stacking sequence $[\theta_1\, \theta_4\, \theta_2\, \theta_3]$. The last ply is then inserted in position $\Xi_2 = 5$ to obtain the guide laminate $[\theta_1\, \theta_4\, \theta_2\, \theta_3\, \theta_5]$. Note that $[\Xi_1\, \Xi_2]$ can have the same value, in this case θ_4 would be inserted first before being shifted one position in order to let θ_5 be in its exact position.

The same principle of insertion is used for the other genotype fields. If the balanced and 10% rule design guideline are active then the thinnest ply is obtained as a combination of theta's and inserted balanced pairs and 10% rule

angles. The order in which insertion occurs during the decoding is: Balanced - 10% rule - and Ply Insertion.

4.2 Stiffness Matrices

For stiffness-based optimisation the [A], [B], and [D] stiffness matrices are obtained from stacking sequences as follows:

In-plane	Coupled	Out-of-Plane
$[A] = \sum_{i=1}^{N} \bar{D}_i(z_i - z_{i-1})$	$[B] = \frac{1}{2}\sum_{i=1}^{N} \bar{D}_i(z_i^2 - z_{i-1}^2)$	$[D] = \frac{1}{3}\sum_{i=1}^{N} \bar{D}_i(z_i^3 - z_{i-1}^3)$

with, $(z_i = -h/2 + t_{ply} \times i)$ is the location of ply i in the laminate, $(h = t_{ply} \times N)$ is the laminate thickness and $\bar{D}_i$ denotes the transformed stiffness matrix of ply i. The total number of plies in the laminate is denoted by N. Calculation of ply stiffness and compliance matrices are easily accessible in the literature, for instance see [**?**]. Additionally, it should be noted that the linear matrix indexing used in MATLAB to refer to matrix data is illustrated in Figure 1.2.

1	4	7
2	5	8
3	6	9

(a) Linear

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

(a) Subscript

Figure 1.2: 3x3 Matrix Indexing

4.3 Lamination Parameters

For lamination parameters-based optimisation stacking sequences are converted into lamination parameters using the following notation:

$$\boldsymbol{LP} = [V_1^A\, V_2^A\, V_3^A\, V_4^A\,,\, V_1^B\, V_2^B\, V_3^B\, V_4^B\,,\, V_1^D\, V_2^D\, V_3^D\, V_4^D]^T \tag{1.1}$$

In-plane	Coupled	Out-of-Plane
$V_1^A = \frac{1}{N}\sum_{i=1}^{N}\cos(2\theta_i)$	$V_1^B = \frac{2}{N^2}\sum_{i=1}^{N}(Z_i^2 - Z_{i-1}^2)\cos(2\theta_i)$	$V_1^D = \frac{4}{N^3}\sum_{i=1}^{N}(Z_i^3 - Z_{i-1}^3)\cos(2\theta_i)$
$V_2^A = \frac{1}{N}\sum_{i=1}^{N}\sin(2\theta_i)$	$V_2^B = \frac{2}{N^2}\sum_{i=1}^{N}(Z_i^2 - Z_{i-1}^2)\sin(2\theta_i)$	$V_2^D = \frac{4}{N^3}\sum_{i=1}^{N}(Z_i^3 - Z_{i-1}^3)\sin(2\theta_i)$
$V_3^A = \frac{1}{N}\sum_{i=1}^{N}\cos(4\theta_i)$	$V_3^B = \frac{2}{N^2}\sum_{i=1}^{N}(Z_i^2 - Z_{i-1}^2)\cos(4\theta_i)$	$V_3^D = \frac{4}{N^3}\sum_{i=1}^{N}(Z_i^3 - Z_{i-1}^3)\cos(4\theta_i)$
$V_4^A = \frac{1}{N}\sum_{i=1}^{N}\sin(4\theta_i)$	$V_4^B = \frac{2}{N^2}\sum_{i=1}^{N}(Z_i^2 - Z_{i-1}^2)\sin(4\theta_i)$	$V_4^D = \frac{4}{N^3}\sum_{i=1}^{N}(Z_i^3 - Z_{i-1}^3)\sin(4\theta_i)$

$Z_i = -N/2 + i$ with N is the number of plies in the laminate and θ_i is the fibre angle of ply i.

5 Fitness Functions

The default fitness functions provided with the toolbox are presented in this section. Evaluating the quality of potential solution is critical to the success of the optimisation, however there is no clear 'better' method for doing so. Only a few of the available index of matching quality are pre-coded into **Opti-BLESS**. Those are:

- The root mean square error (RMSE)
- The mean norm error (NormE)
- The absolute maximum error (MaxAE)

In theory, computing multiple quality indexes ensure that the robustness of the solutions. It may, therefore, be conceivable to combine some of the index mentioned above. In practice, finding adequate weighting factor between the different quality indexes is not straightforward. In general there will be no obvious answers because each problem is unique and it is recommended that you experiment various fitness functions in order to find the one best suited to your case.

5.1 Lamination Parameters

This section deals with fitness function based on lamination parameters. One of the default fitness function in **Opti-BLESS** is based on the root mean square error ($RMSE$) between lamination parameters given as input and the lamination parameters retrieved by the GA:

$$\min(Fitness) = \min\left(\frac{1}{N_{lam}}\sum_{p=1}^{N_{lam}} RMSE_p\right) \tag{1.2}$$

where, the root mean square error is simply expressed as:

$$RMSE_p = \sqrt{\frac{1}{12}\sum_{i=1}^{12}\left[\boldsymbol{\alpha}_i\left(\widetilde{\boldsymbol{LP}}_{i,\,p} - \boldsymbol{LP}_{i,\,p}\right)\right]^2} \tag{1.3}$$

where, $\widetilde{\boldsymbol{LP}}_{i,\,p}$ is the vector of input lamination parameters for laminate p and $\boldsymbol{LP}_{i,\,p}$ is the vector of lamination parameters obtained by the GA. It is possible to emphasis the optimisation based on matching specific lamination parameters. For instance, you may be interested in in-plane parameters matching more that the out-of-plane parameter matching and this should reflect in the fitness evaluation function. This is done with the help of a vector of scaling coefficients $\boldsymbol{\alpha}$.

5.2 Stiffness Matrices

This section deals with fitness function based on stiffness parameters. One of the default fitness function in **Opti-BLESS** is based on the average of the RMS error between the stiffness matrices given as input and the retrieved one by the GA:

$$\min(Fitness) = \min\left(\frac{1}{N_{lam}}\sum_{p=1}^{N_{lam}}\Big(RMSE_p[A] + RMSE_p[B] + RMSE_p[D]\Big)\right) \tag{1.4}$$

$$RMSE_p[A] = \sqrt{\frac{1}{9}\sum_{i=1}^{9}\left[\boldsymbol{\alpha}_i\left([\tilde{A}]_{i,\,p} - [A]_{i,\,p}\right)\right]^2} \tag{1.5}$$

where, p is the patch number.

6 Tutorials

In the main toolbox folder you will find various introductory files entitled *Learn2use_xxx* which will help you get started using **Opti-BLESS**. While these are self-explanatory files, explanation for the first file are also given in this section.

6.1 First Example - Single Patch Lamination Parameter Optimisation

You should first start with *Learn2use_Example1.m*. In this example, the lamination parameters of a simple stacking sequence have been precomputed and we will use **Opti-BLESS** in order to retrieve a matching stacking sequence. Ideally, the exact same stacking sequence:

SS = [Bottom ply / ... / Top ply]
SS = [45/-45/90/0/45/90/0/45]

Table 1.6: Lamination parameters of the 8-ply
SS = [45/-45/90/0/45/90/0/45]

LP2Match	
0	V_1^A
0.25	V_2^A
0	V_3^A
0	V_4^A
0.125	V_1^B
0.1875	V_2^B
0.25	V_3^B
0	V_4^B
0.046875	V_1^D
0.4375	V_2^D
-0.46875	V_3^D
0	V_4^D

```
%% === Objective

% --- Staking Sequence used as example,         - Bottom ply [45/-45/90/0/45/90/0/45] Top ply
% --- Laminate paramaters are obtained using:   - Convert_SS2LP([ 45   -45    90     0    45    90     0    45])
Lp2Match = [
             0   % V1A
          0.25   % V2A
             0   % V3A
             0   % V4A
         0.125   % V1B
        0.1875   % V2B
          0.25   % V3B
             0   % V4B
      0.046875   % V1D
        0.4375   % V2D
      -0.46875   % V3D
             0];  % V4D


Objectives.UserFct = false;        % set to false to used one of the pre-defined fitness functions
Objectives.Type    = 'LP';         % Set the objective function to lamination parmeter based

ScalingCoef  = ones(12,1);  % Set the relative importance to all lamination parameters (equal in this example)

% --- construct the objective Table
%                        Patch ID           Lower and upper
%                                        number of plies boundary
Objectives.Table   = [{'Laminate #'}          {'Nplies'}            {'LP2Match'}     {'Scaling Coefficient'} ;
                              {1}               {[8 8]}            {Lp2Match(:,1)}         {ScalingCoef}    ];

Objectives.FitnessFct = @(LP) RMSE_LP(LP,Objectives);       % Attribute the function handle used to calculate fi
```

```
%% === Design Guidelines
%                       [Symmetry,  Balanced,  Damtol,   Rule10percent,  Disorientation,  Contiguity,  InternalContin
Constraints.Vector     = [false    ,    false ,  false ,      false     ,     false     ,     false  ,      false
Constraints.DeltaAngle = 45;

Constraints.NContiguity   = 3;  % optional (only needed if Contiguity = true)
Constraints.NInternalCont = 3;  % optional (only needed if InternalContinuity = true)
```

```
%% === Options
GAoptions.Npop    = 20;                    % Population size
GAoptions.Ngen    = 100;                   % Number of generations
GAoptions.NgenMin = 100;                   % Minimum number of generation calculated
GAoptions.Elitism = 0.01;                  % Percentage of elite passing to the next Gen. (from 0 to 1)
GAoptions.PC      = 0.75;                  % Percentage of crossover (from 0.1 to 1)
GAoptions.IniPopFEASIBLE = 1;              % Either (1 or 2), Ensure the initial population 1:respect all design guid

GAoptions.FitnessLimit = 1e-5;             % Value at which the GA will stop if a fitness is found below this thresho
GAoptions.PlotInterval = [10];             % Refresh plot every X itterations
GAoptions.SaveInterval = [2];              % Save Data every    X itterations (in Results.txt)
GAoptions.PlotFct      = @gaplotbestf;     % Refresh plot every X itterations
GAoptions.OutputFct    = @GACustomOutput;  % Custom ouput function (can be changed to output any information regardin
```

```
%% === Run
[Output]  = OptiBLESS(Objectives,Constraints,GAoptions);

display(Output)
display(Output.Table)
```

Figure 1.4: Example 1 - Objective Structure

The objective structure, containing the lamination parameters to be matched, is constructed as shown in Figure 1.5. The type of objective function is defined first, a lamination parameter matching strategy in this case (i.e. LP). The scaling coefficient is defined next. It influences the fitness evaluation by scaling up or down specific lamination parameter matching errors. In our example, we wish to match all lamination parameters equally well and a 12 vector

of ones is therefore used. Next, the 4 fields of the objective table are filled. This includes a patch ID, the lower and upper number of plies allowed, the lamination parameters to be matched and the scaling coefficient for this patch. Finally, the fitness evaluation function used to evaluate the performance of GA individuals is defined, a RMS error fitness function called *RMSE_LP* is used for this purpose.

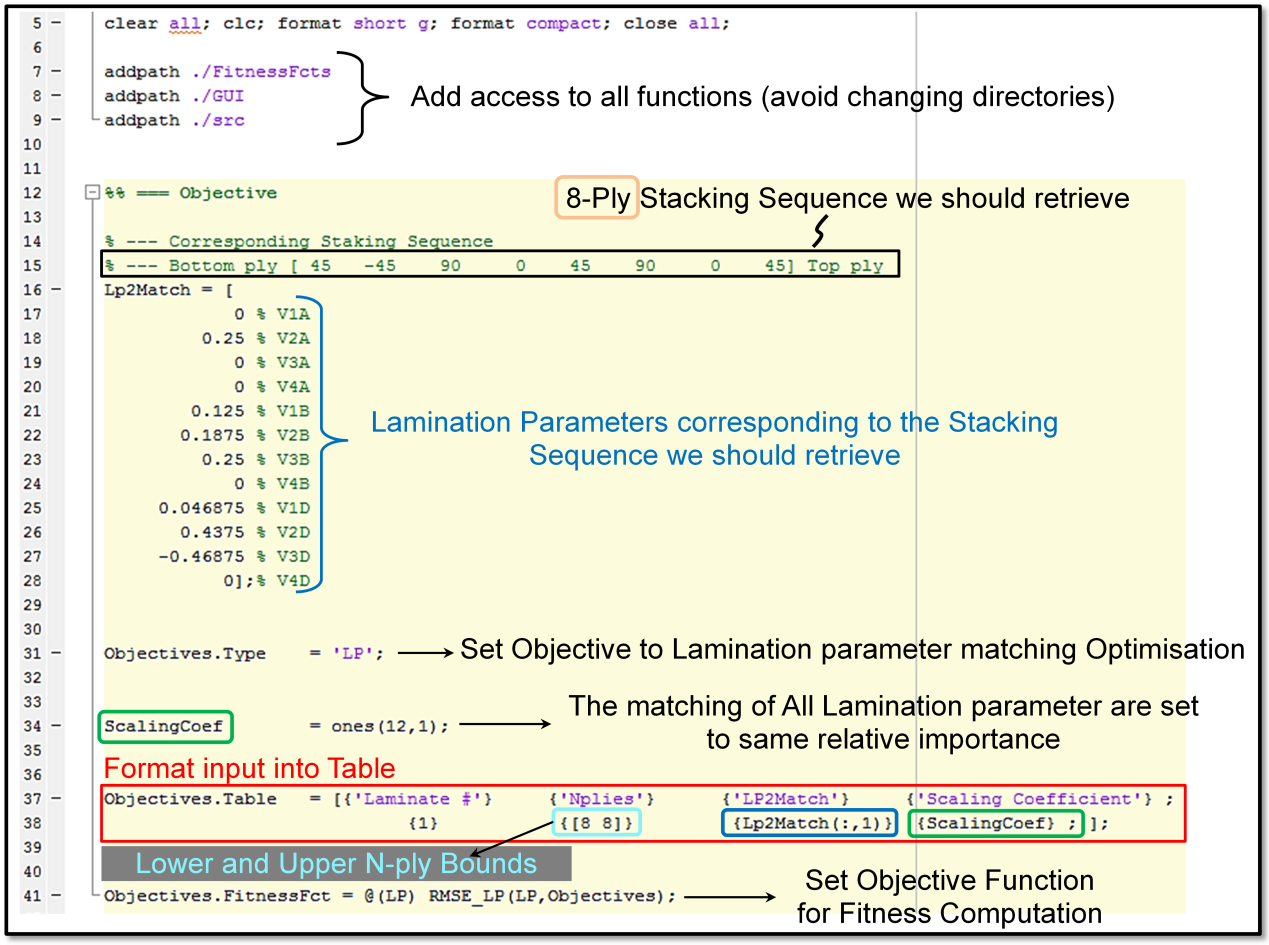


Figure 1.5: Example 1 - Objective Structure

The constraints structure is next. First, a vector of boolean is used to activate or deactivate the various constraints available. The discrete step angles defining the size of the design space are then defined. Our stacking sequence example is neither balanced nor symmetric and therefore we set the Boolean balanced and Sym to false. The ORDERED boolean value is not used in the case of single laminates and can be set to either true or false. The GA Options structure is then the last one to define before running the optimisation as shown in Figure 1.7.

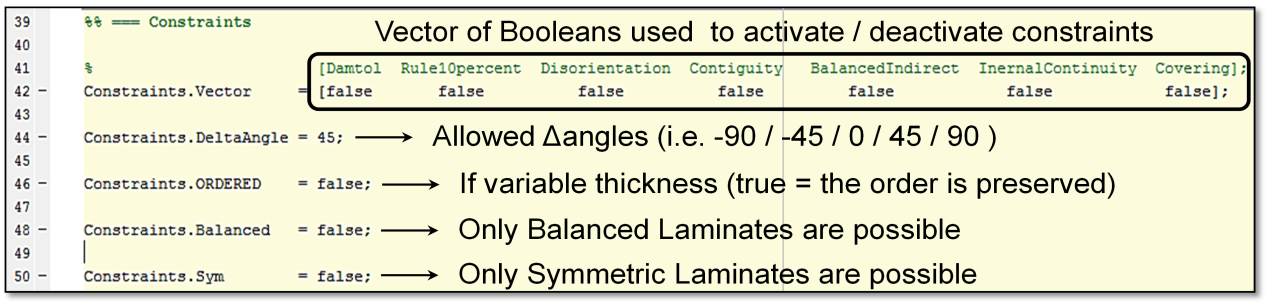


Figure 1.6: Example 1 - Constraint Structure

```
%% === Options
GAoptions.Npop     = 100;                  % Population size
GAoptions.Ngen     = 100;                  % Number of generations
GAoptions.NgenMin = 100;                   % Minimum number of generation calculated
GAoptions.Elitism = 0.01;                  % Percentage of elite passing to the next Gen. ([0 1])
GAoptions.PC       = 0.75;                 % Percentage of crossover ([0 1])
GAoptions.PlotInterval = [10];             % Refresh plot every X itterations (leave empty [ ] to suppres plot)
GAoptions.SaveInterval = [2];              % Save Data every X itterations    (leave empty [ ] to suppres saves)
GAoptions.PlotFct      = @gaplotbestf;     % Default MATLAB Plot function (Not distributed with the toolbox due to Copyright)
GAoptions.OutputFct    = @GACustomOutput;  % Default Output function
```

Figure 1.7: Example 1 - Option Structure (self explanatory)

Once the three input structures have been defined, the **Opti-BLESS** optimisation is run by calling **OptiBLESS** as in Figure 1.8. Make sure that you have the GAOptions.PlotInterval not empty in order to plot the evolution of fitness. The results of the optimisation are returned into the Output structure. Check the structure field called **Table** to find a summary of the results including matching quality, the retrieved stacking sequence and corresponding lamination parameters. The final best found results are then visualised by calling the plot function.

```
%% === Run
[Output]  = OptiBLESS(Objectives,Constraints,GAoptions);   ——— Run the toolbox and return the output structure

display(Output)
display(Output.Table)   ——— Results Display

%% === Plot
plotSS(Output)   ——— Results plot, geometry can be added as optional input
```

Figure 1.8: Example 1 - Run and Results Visualisation

If everything has worked so far and the code has plotted results similar to the one presented in Figures 1.9 and 1.10 you are on the right track. A text file called *results.txt* should also have been created in your main folder containing the summary of best and average fitness values every 2 generations as specified by the *GAOptions.SaveInterval* value. You have now learnt to set an input file and run **Opti-BLESS**. You will see that more advanced problems are not really much more difficult to set up.

Figure 1.9: Example 1 - First Output

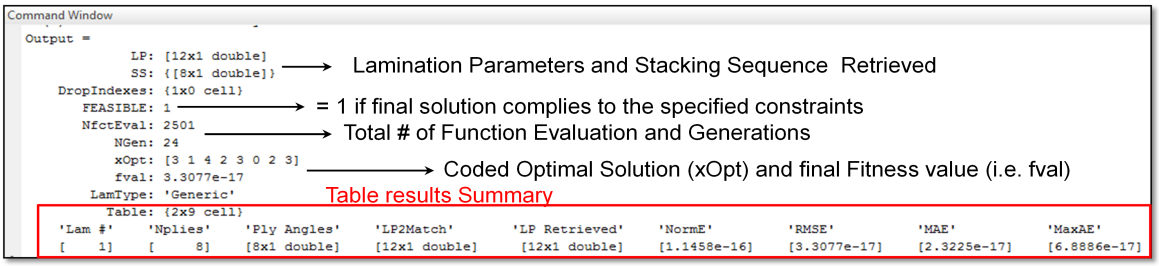


Figure 1.10: Example 1 - Second Output

7 Concluding Remarks

The **Opti-BLESS** toolbox is a free open-source project for the optimisation of composite stacking sequence developed in MATLAB. In its current form the toolbox has been validated and has shown to perform well even for advanced blending problems.

In my current position the development of the **Opti-BLESS** toolbox is only a free-time side project and much interesting features and idea remains to be tested. In particular, a composite optimisation tool that would significantly reduce the discrepancies between optimised and manufacturable design comes to mind. Of course everyone is welcome to simply use and/or contribute to the toolbox development. Feedback is also always appreciated. Below is a list of the current limitations that I have identified and could be a basis for future work.

7.1 Current Limitations

The current limitations of **Opti-BLESS** include:

- The curse of dimensionality. Despite the concise coding provided by the guide-based approach, the growth of design variables will quickly limit the capability of the genetic algorithm in finding an optimal solution.
- Limited geometrical capabilities. The structure geometry could be used in future release so as to have a more significant influence on the final optimised design. For instance, by considering by drop rates, gap and overlap due to automated fibre placement.
- Transition section between patches are considered negligible during fitness calculation. Intermediate sections contain all ply drops and are therefore critical parts of the structure, due to stress concentration and trough-thickness load distribution, where failure is likely to start.
- Tow-steering methodologies are not considered. Only patch designs are possible.
- A single material type is used. All plies are made of the materials. This limitation could be easily removed in future release.
- The addition of detailed manufacturability constraints could further help bridging the gap between optimised and manufacturable designs.

Bibliography

[1] David B Adams, Layne T Watson, Zafer Gürdal, and Christine M Anderson-Cook. Genetic algorithm optimization and blending of composite laminates by locally reducing laminate thickness. *Advances in Engineering Software*, 35(1):35–43, 2004.

[2] Terence Macquart, Noud Werter, and Roeland De Breuker. Aeroelastic tailoring of blended composite structures using lamination parameters. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1966, 2016.

[3] Terence Macquart, Marco T Bordogna, Paul Lancelot, and Roeland De Breuker. Derivation and application of blending constraints in lamination parameter space for composite optimisation. *Composite Structures*, 135:224–235, 2016.

[4] Mostafa M Abdalla, Christos Kassapoglou, and Zafer Gürdal. Formulation of composite laminate robustness constraint in lamination parameters space. In *Proceedings of 50th AIAA/ASME/ASCE/AHS/ASC Conference*, 2009.

[5] F-X Irisarri, Alexis Lasseigne, F-H Leroy, and Rodolphe Le Riche. Optimal design of laminated composite structures with ply drops using stacking sequence tables. *Composite Structures*, 107(1):559–569, 2014.