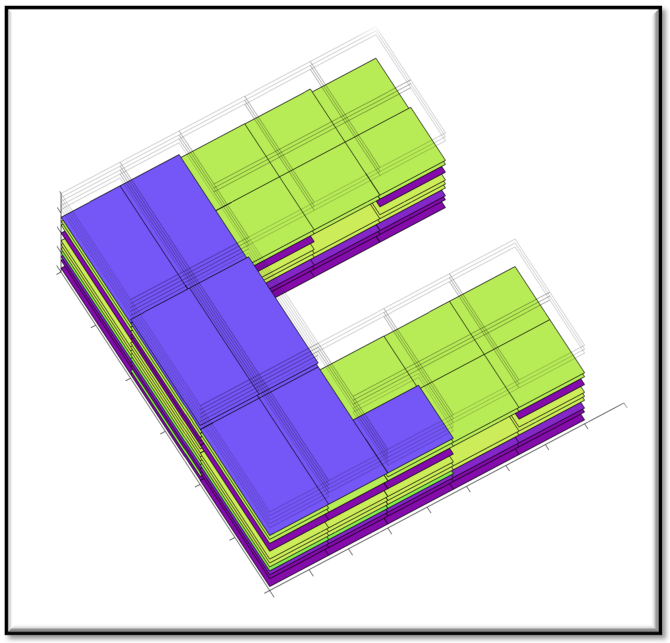
SSORT - A Free Open-source Stacking Sequence Optimisation and Retrieval Toolbox (MATLAB)

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 remains closed proprietary codes. By contrasts, the toolbox presented is this report provides a first step towards the development of an open-source free tool for the optimisation of composite structures. The Stacking Sequence Optimisation and Retrieval Toolbox (**SSORT**) is a small and easy-to-use code for the optimisation of composite structures. In simple terms, **SSORT** 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 **SSORT** and provides capability overviews and procedures for software execution, as well as a variety of example studies.

Preface

The **SSORT** 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 have been proposed for composite optimisation, these approaches were not easily accessible. I, therefore, decided to write a piece of code that others could use instead of having to write their own. This document serves as a user introduction to the Stacking Sequence Optimisation and Retrieval Toolbox (**SSORT**) developed for MATLAB.

Requirement
MATLAB 2014.b or more recent
MATLAB Optimisation Toolbox
SSORT Toolbox (ADD LINK)

Contact Information:
TMacquart.ssort@gmail.com

Nomenclature

$\boldsymbol{N}$	=	Number of plies per laminate
$\boldsymbol{\theta}$	=	Guide laminate ply angles
$\boldsymbol{L}$	=	Balanced ply angle pairs locations in the laminate
$\boldsymbol{\Xi}$	=	Guide drops
$\boldsymbol{\alpha}$	=	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
$LP2Match$	=	Lamination parameters defined as input to be matched by **SSORT**

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 unique laminate properties such as ply angles and the number of plies. Using unique laminate properties as design variables allows manufacturing constraints to easily be integrated in the direct optimisation framework. 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. 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 laminate design space.

Based on this short introduction, 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 the authors' point of view (possibly biased) 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, for lamination parameter spaces in particular [1, 2]. However, the difficulties arising in retrieving laminates from the parametrised spaces are still problematic and have motivated the development of the **S**tacking **S**equence **O**ptimisation and **R**etrieval **T**oolbox - **SSORT**.

Nevertheless, it likely that both the direct and indirect frameworks will still be used in the future and as a result **SSORT** also provides direct composite optimisation capabilities.

If you are somewhat already familiar with the conventional notation used for composite materials you can try to skip the following few sections and directly go to the Tutorial section.

2 Toolbox Capabilities

SSORT has the following capabilities:

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

In order to find the best stacking sequences, **SSORT** allows the **laminates' thicknesses**, **fibre angles** and **ply drops** to be optimised using the genetic algorithm provided by MATLAB. Additionally, the following list of design constraints [3] for composites can be used:

List of Constraints

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**. The change of angles between two consecutive plies should not be below 5°.
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 three plies.
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 **SSORT** algorithm. As it can be seen, users have very little data to actually process because most of the code serves as a wrapper around the MATLAB genetic algorithm.

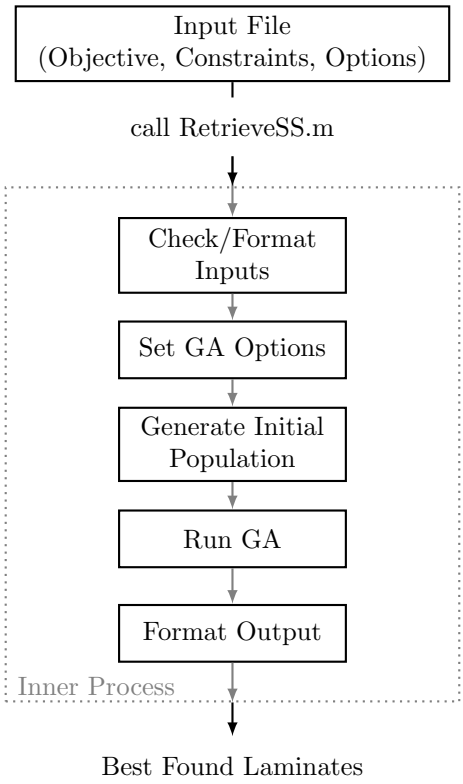


Figure 1.1: **SSORT** algorithm overview

The input file contains three structures. namely objective, constraints and options. Once the three structures are defined, the main function (i.e. **RetrieveSS**) can be 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. Four 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 - Stiffness matrices to be matched $[A]$ $[B]$ and $[D]$ are given as input and converted into their equivalent set of lamination parameters. A lamination parameter matching fitness function is then used to evaluate the fitness of stacking sequences evaluated by the genetic algorithm.

Type 4 - Users can defined their own fitness functions. These can be based on either lamination parameters, stiffness matrices or stacking sequences.

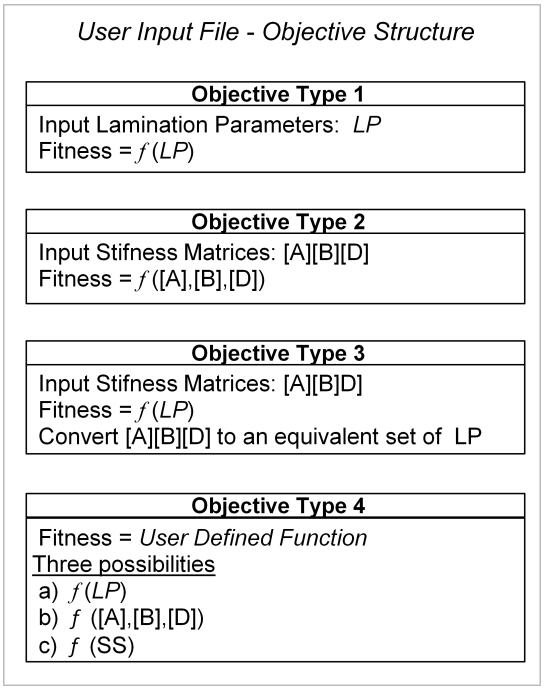


Figure 1.2: The different types of objective structures.

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 1.3.

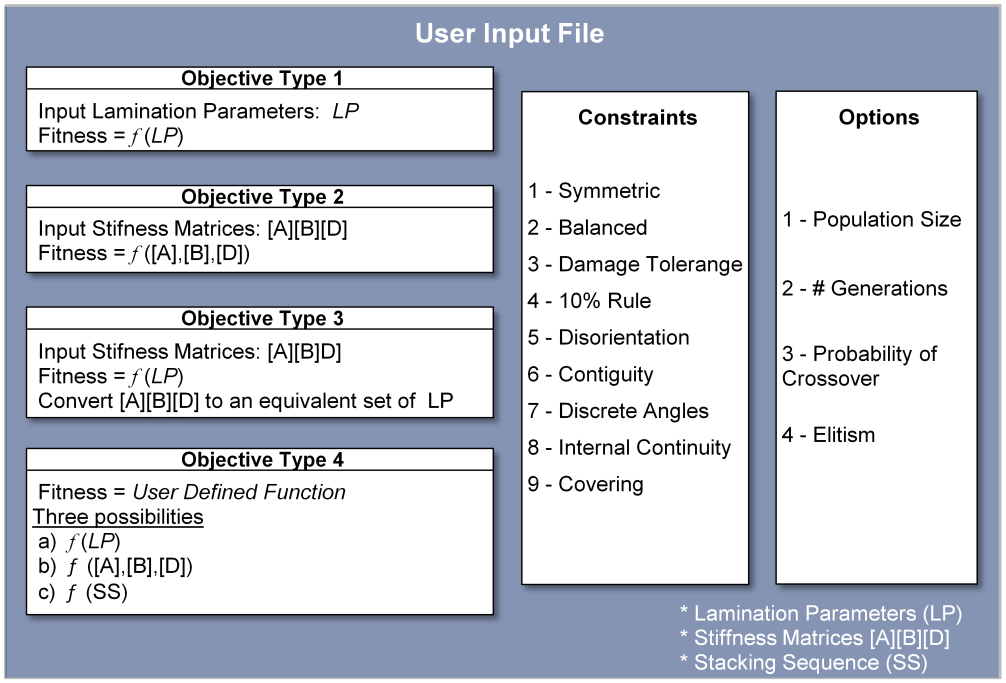


Figure 1.3: **SSORT** input file overview

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. To the best of the author's knowledge there is no comparisons between the various coding method and their impact on the optimisation of composites. The coding methodology used in **SSORT** has been developed by the author and inspired from the guide-based approach to composite optimisation [3, 4]. Each individual is coded based on 4 distinct fields of design variables:

1. Number of plies for each patches (**N**)
2. The guide laminate ply angles ($\boldsymbol{\theta}$)
3. The location of the guide pair angles in the balanced laminate (**L**)
4. The guide ply drops ($\boldsymbol{\Xi}$)

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 **SSORT**.

Table 1.1: Numerical representation of typical solutions in **SSORT**, see Table 1.2 for the corresponding Stacking Sequences ('-' indicate that it is not a design variable)

	Number of Plies	**Guide Fibre Angles**	**Fibre Angle Pair Locations** (only if Balanced)	**Guide Ply-Drops**
Generic Genotype	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°, 90°, 0°, −45°, 45°	-	2, 3
(C)	5, 3	45°, 90°, 0°, −45°, 45°	-	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 1.2: Numerical representation of typical solutions in **SSORT** (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°/−30°/45°/ 60°/−20°/−10°/20°/−45°/−60°/10° Lam_1 = 30°/−30°/ - / 60°/−20°/−10°/20°/ - /−60°/10°

4.2 Stiffness Matrices

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 [5]. Additionally, it should be noted that the linear matrix indexing used in MATLAB to refer to matrix data is illustrated in Figure 1.3.

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 1.3: 3x3 Matrix Indexing

4.3 Lamination Parameters

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 **SSORT**. Those are:

- The root mean square error (RMSE)
- The mean norm error (NormE)
- The absolute maximum error (MaxAE)
- Weighted combinations of the above

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 Matching

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

$$\min(Fitness(\boldsymbol{\theta}, \boldsymbol{L}, \boldsymbol{\Xi}, \boldsymbol{N})) = \min\left(\frac{1}{N_{lam}} \sum_{p=1}^{N_{lam}} RMSE_p(\boldsymbol{\theta}, \boldsymbol{L}, \boldsymbol{\Xi}, \boldsymbol{N})\right) \tag{1.2}$$

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

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. In particular cases you may emphasis on matching specific lamination parameters. For instance, you may values an in-plane parameters match 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 scaling coefficient $\boldsymbol{\alpha}$.

5.2 Stiffness Matrices Matching

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

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

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

where, p is the patch number, $\boldsymbol{\theta}$ denotes the guide staking sequence fibre angles, $\boldsymbol{\Xi}$ are the guide ply drop locations, and the number of plies are stored in $\boldsymbol{N}$. The linear matrix indexing used in MATLAB is illustrated in Figure 1.3.

6 Tutorial - Learn2use

In the main toolbox folder you will find various introductory files entitled *Learn2use_xxx* which will help you get started using **SSORT**. Explanation for each of these files are 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 **SSORT** in order to retrieve a matching stacking sequence. In the best cases, the exact same matching sequence:

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

Table 1.5: 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

The objective structure, containing the lamination parameters to be matched, is constructed as shown in Figure 1.4. 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.

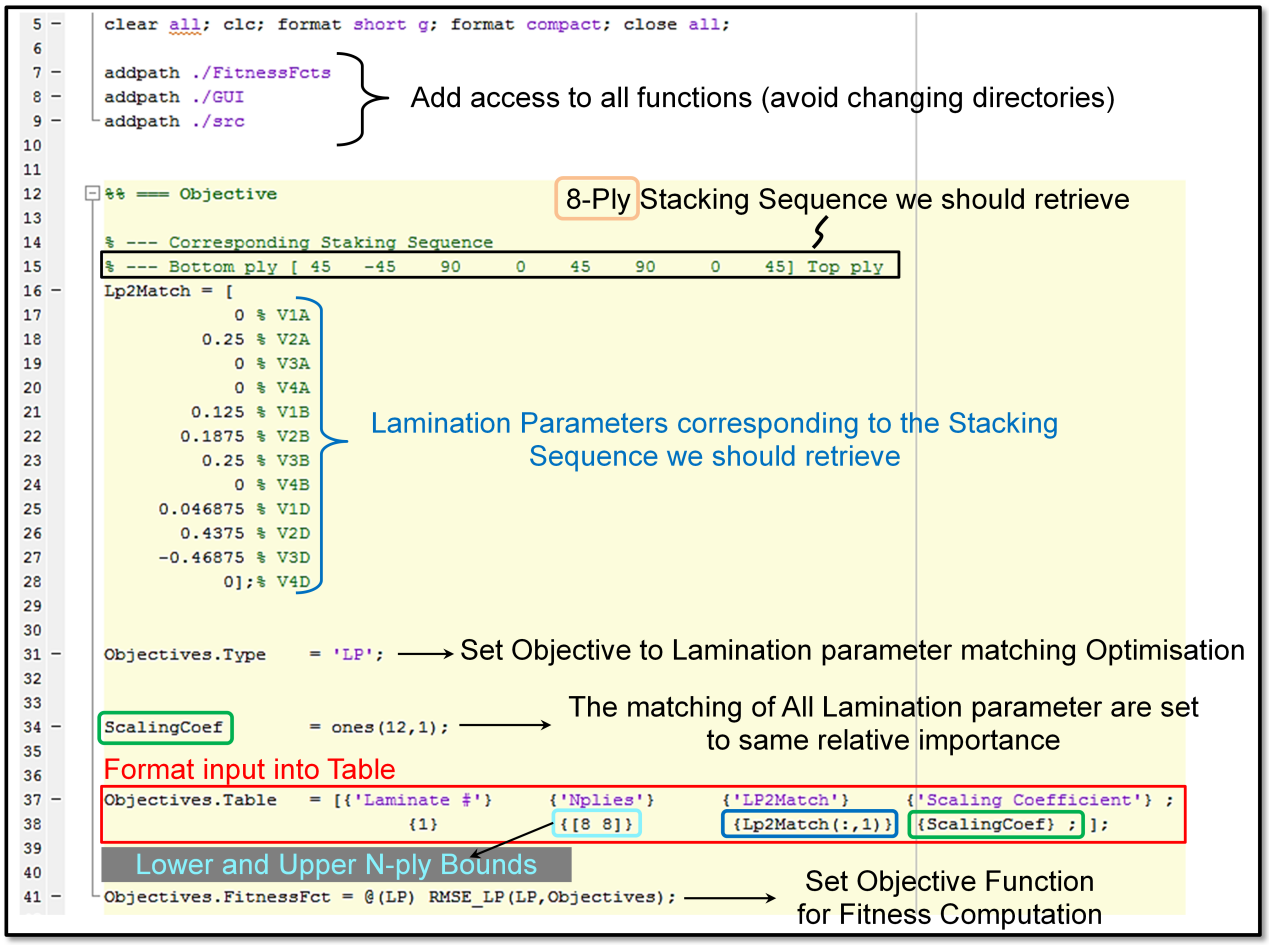


Figure 1.4: Example 1 - Objective Structure

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 1.6.

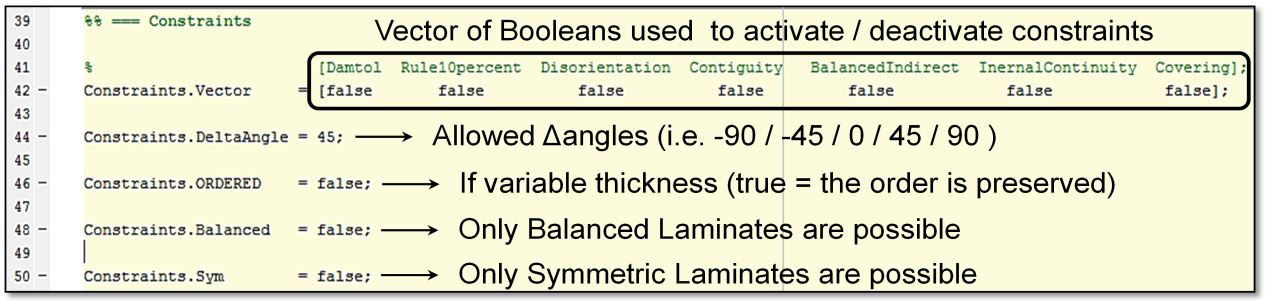


Figure 1.5: Example 1 - Constraint Structure

```
%% === 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 1.6: Example 1 - Option Structure (self explanatory)

Once the three input structures have been defined, the **SSORT** optimisation is run by calling **Retrieve_SS** as in Figure 1.7. 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]  = RetrieveSS(Objectives,Constraints,GAoptions);   --> Run the Toolbox, Returns the output result structure

display(Output)
display(Output.Table)   --> Display results


%% === Plot
plotSS(Output)   --> Plot results (the structure geometry can be added as optional input)
```

Figure 1.7: 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 1.8 and 1.9 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 **SSORT**. You will see that more advanced problems are not really much more difficult to set up.

Figure 1.8: Example 1 - First Output

```
Command Window
Output =
           LP: [12x1 double]
           SS: {[8x1 double]}   --> Lamination Parameters and Stacking Sequence  Retrieved
  DropIndexes: {1x0 cell}
     FEASIBLE: 1   --> = 1 if final solution complies to the specified constraints
     NfctEval: 2501   --> Total # of Function Evaluation and Generations
         NGen: 24
         xOpt: [3 1 4 2 3 0 2 3]   --> Coded Optimal Solution (xOpt) and final Fitness value (i.e. fval)
         fval: 3.3077e-17
      LamType: 'Generic'
        Table: {2x9 cell}   Table results Summary
  'Lam #'    'Nplies'    'Ply Angles'    'LP2Match'       'LP Retrieved'    'NormE'          'RMSE'           'MAE'            'MaxAE'
  [    1]    [     8]    [8x1 double]    [12x1 double]    [12x1 double]     [1.1458e-16]     [3.3077e-17]     [2.3225e-17]     [6.8886e-17]
```

Figure 1.9: Example 1 - Second Output

6.2 Second Example - Single Patch Stiffness Optimisation

We are now moving onto the second example entitled *Learn2use_Example2.m*. In this example things are a bit different because we are going to optimise a stacking sequence in order to match stiffness properties, by contrast to lamination parameters. For sake of simplicity, the same 8-ply stacking sequence is used. This time the input file contains the stacking sequence equivalent stiffness matrices and the objective table is slightly different as can been seen in Figure 1.10.

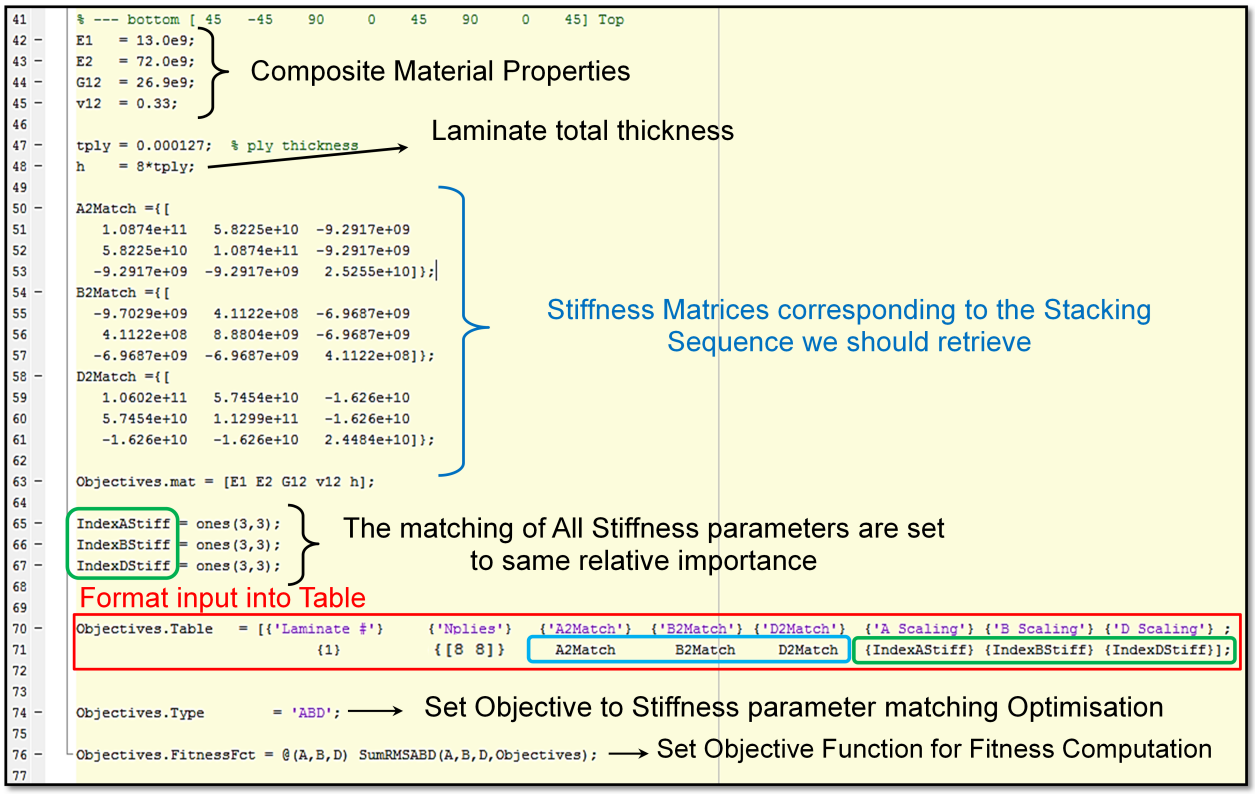


Figure 1.10: Example 2 - Objective Structure

For the other input structures, runs and plot, all should be very similar to the first example and self-explanatory.

6.3 Third Example - Multi-patch Optimisation

Things are starting to get more interesting in this example (*Learn2use_Example3.m*). We now wish to retrieve the lamination parameters corresponding to a 3-patch balanced composite structure. A guide laminate and 2 thinner laminates:

SS = [Bottom ply (1) / ... / Top ply (10)]
SS_{guide} = [-45/0/45/90/0/-45/45/90/-45/45]
Lam_1 = [0 / 90/ 0/ -45/ 45/ 90/ -45/ 45] - Removed plies [1 and 3]
Lam_2 = [0 / 90/ 0/ 90/ -45/ 45] - Removed plies [1, 3, 6 and 7]

Although this example is relatively simple, you will not always retrieve the exact optimal stacking sequences. This shows that stacking sequence retrieval using evolutionary algorithms is only an approximate process. If you have retrieved the exact solution then the toolbox should have results in something similar to Figure 1.11. If you have been unlucky you may try to rerun the optimisation or increasing population size until you find the exact optimal solution. Note that since no geometry was given in the plot input, the composite patches are simply plotted next to each other in Figure 1.11.

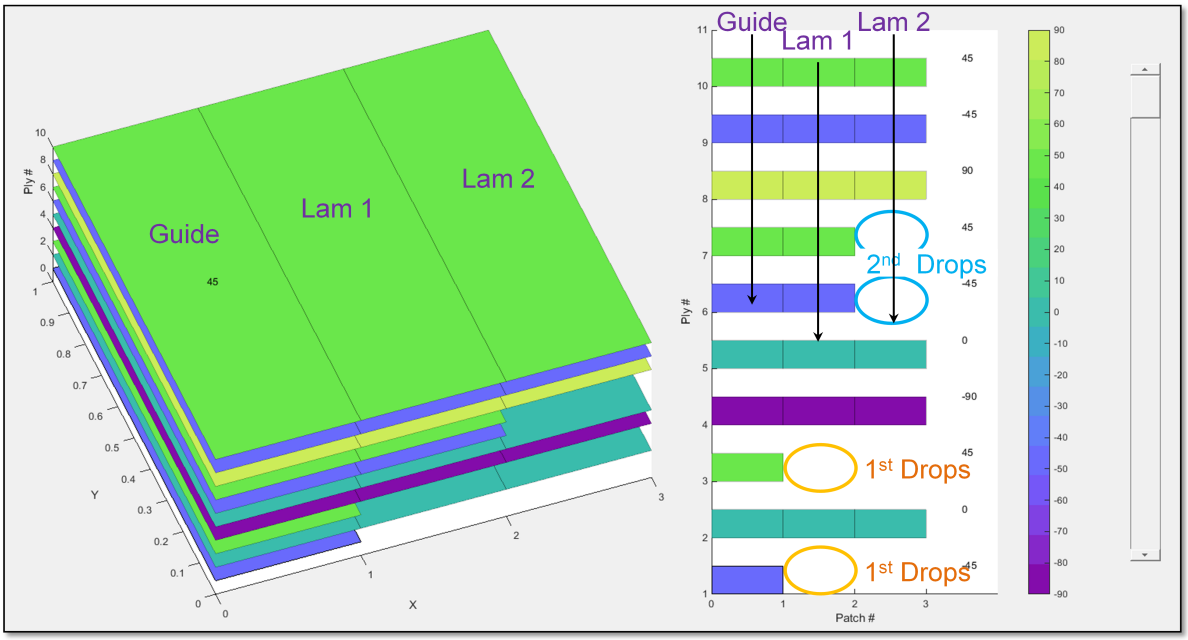


Figure 1.11: Example 3 - Output for the exact matching retrieved result

Let us now try to match lamination parameters without the exact knowledge of the number of plies required. That is, for composite optimisation the number of plies is often a design variable. In **SSORT** you simply need to specify the boundaries around which the number of plies are allowed to vary as shown in Figure 1.12 , the toolbox will do the rest. As you will notice, optimising with variable number of plies might decrease the quality of the results. A few key things are actually changing inside **SSORT** when the number of plies are used as design variables. For instance, the genotype use to code individual solutions will contains more design variables and will also likely include non-coding genes. By default the fitness functions use in **SSORT** do not penalise an increase in thickness, if you wish to do so you can simply create your own fitness function based on the default one provided.

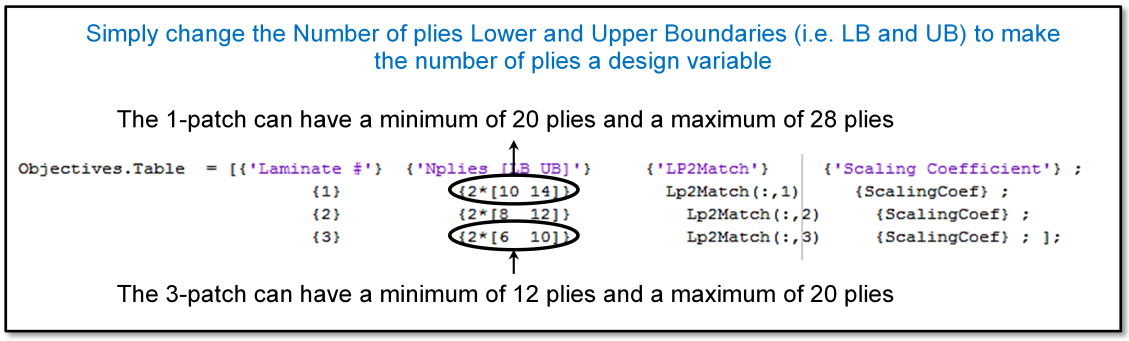


Figure 1.12: Example 3 - Modification for variables number of plies

6.4 Fourth Example - User Fitness Function

All the fitness function used in previous examples are default evaluation functions that have been defined beforehand. These are all stored in the *FitnessFcts* folder. This section will explain how you can define and use your own fitness function.

6.5 18-Panel Horseshoe Benchmark Problem (Direct Optimisation)

7 Concluding Remarks

7.1 Current Limitations

1. For balanced laminates, all ply angles have an opposite pair including 90 and 0 degree plies. This does not have to be the case.

2. The composite structure geometry does not influence the retrieved stacking sequence and is only used for plots. This is definitely something that must be addressed in order to improve the manufacturability of the optimised designs.

3. The current genetic algorithm used is generic, more advanced evolutionary algorithms and methods exists that could improve the search performance. Fortunately, it should not require significant efforts to change the optimiser.

Bibliography

[1] Abdalla, M. M., Kassapoglou, C., and Gürdal, Z., "Formulation of composite laminate robustness constraint in lamination parameters space," *Proceedings of 50th AIAA/ASME/ASCE/AHS/ASC Conference*, 2009.

[2] Macquart, T., Bordogna, M. T., Lancelot, P., and De Breuker, R., "Derivation and application of blending constraints in lamination parameter space for composite optimisation," *Composite Structures*, Vol. 135, 2016, pp. 224–235.

[3] Irisarri, F.-X., Lasseigne, A., Leroy, F.-H., and Le Riche, R., "Optimal design of laminated composite structures with ply drops using stacking sequence tables," *Composite Structures*, Vol. 107, 2014, pp. 559–569.

[4] Adams, D. B., Watson, L. T., Gürdal, Z., and Anderson-Cook, C. M., "Genetic algorithm optimization and blending of composite laminates by locally reducing laminate thickness," *Advances in Engineering Software*, Vol. 35, No. 1, 2004, pp. 35–43.

[5] Roylance, D., "Laminated composite plates," *Massachusetts Institute of Technology Cambridge*, 2000.