BIANCA, A Genetic Algorithm for Engineering Optimisation - User guide

Marco Montemurro, Paolo Vannucci, Angela Vincenti

BIANCA, A Genetic Algorithm for Engineering Optimisation

Version 3.1 User's guide

M. Montemurro

Institut d'Alembert UMR7190 CNRS -Université Pierre et Marie Curie Paris 6, Case 162, 4, Place Jussieu, 75252 Paris Cedex 05, France.

P. Vannucci

Université de Versailles et St Quentin, 45 Avenue des Etats-Unis, 78035 Versailles, France and Institut d'Alembert UMR7190 CNRS -Université Pierre et Marie Curie Paris 6, Case 162, 4, Place Jussieu, 75252 Paris Cedex 05, France.

A. Vincenti

Institut d'Alembert UMR7190 CNRS - Université Pierre et Marie Curie Paris 6, Case 162, 4, Place Jussieu, 75252 Paris Cedex 05, France.

Contents

Chapter 1

Introduction to BIANCA

1.1 Capabilities of BIANCA

The genetic algorithm (GA) BIANCA 3.1 is a multi-population GA able to deal and solve constrained and unconstrained hard combinatorial optimisation problems in engineering. The effectiveness and robustness of BIANCA reside upon the generality and richness in the representation of the information, and on the way the information is extensively exploited during genetic operations. For more details see [1, 2].

In its previous version, BIANCA was a GA substantially based on the principles of the standard GA, see Fig. 1.1. Nowadays, BIANCA is a powerful modular numerical tool, able to deal with general problems of engineering optimisation. The version 3.1 of BIANCA tool-kit provides a flexible, extensive interface between the user simulation code and a variety of iterative methods and strategies.

Moreover, the version 3.1 of the code goes beyond the structure of the standard GA: this version, in fact, is able to let evolve individuals and species at the same time: this is an important feature that makes BIANCA able to deal with optimisation problems concerning modular systems/structures, such as composite plates or shells, stiffened panels and so on. Optimising modular systems with variable number of modules using a GA corresponds to the evolution of a population where individuals/points belonging to different species are mixed. The most part of standard GAs are not able to deal with such problems. For a deeper insight the matter see [3, 4].
BIANCA is written in FORTRAN language.

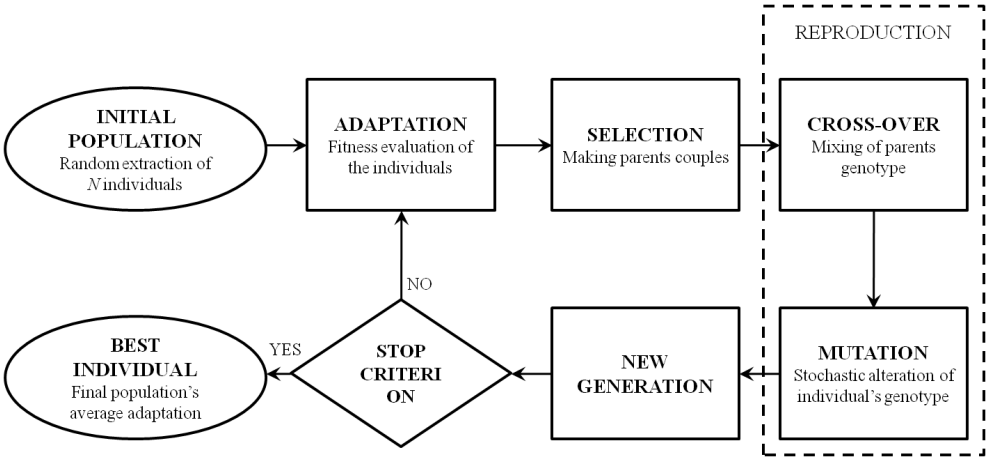

Figure 1.1: Standard GA's scheme.

1.2 Background and mathematical formulations

A general optimization problem is formulated as follows:

$$\text{subject to}: \begin{cases} \min\limits_{\mathbf{x}} f(\mathbf{x}) \\ g_i(\mathbf{x}) \leq 0 \qquad i = 1, ..., r \\ h_j(\mathbf{x}) = 0 \qquad j = 1, ..., m \\ \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \end{cases} \tag{1.1}$$

where vectors and matrix terms are marked in bold typeface. In this formulation $\mathbf{x}$ is the n-dimensional *vector of design variables*, while $\mathbf{x}_L$ and $\mathbf{x}_U$ are the n-dimensional vectors representing the lower and upper bounds of the design variables, i.e. the *design space*. Design variables can be of different type: continuous, regular discrete, scattered discrete or grouped.
The optimisation goal is to minimize the objective function $f(\mathbf{x})$ subject to a given number of constraints: $g_i(\mathbf{x})$ is the r-dimensional vector of inequality constraints, while $h_j(\mathbf{x})$ is the m-dimensional vector of equality constraints.

The optimization problem type can be characterized both by the types of constraints present in the problem and by the linearity or non-linearity of the objective and constraint functions. A problem where at least some of the objective and constraint functions are non-linear is called a non-linear programming (NLPP) problem. These NLPP problems predominate in engineering applications and are the primary focus of BIANCA 3.0.

In BIANCA the equality and inequality constraints are treated by means of a particular strategy which is based on the combination between classi-

cal penalisation methods and the exploitation of the distributed information over the population along the generations. The name of this technique is ADP, which stands for *Automatic Dynamic Penalisation*.
Classical penalisation methods transform Eq.(1.1) into an unconstrained optimisation problem through the definition of a new modified objective function $F(\mathbf{x})$:

$$\min_{\mathbf{x}} F(\mathbf{x})$$

$$F(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & if \quad g_k(\mathbf{x}) \leq 0 \quad k = 1, ..., r \\ & and \quad h_j(\mathbf{x}) = 0 \quad j = 1, ..., m \\ f(\mathbf{x}) + \sum_{k=1}^{r} c_k G_k(\mathbf{x}) + \sum_{j=1}^{m} r_j H_j(\mathbf{x}) & if \quad g_k(\mathbf{x}) > 0 \quad k = 1, ..., r \\ & and \quad h_j(\mathbf{x}) \neq 0 \quad j = 1, ..., m \end{cases} \tag{1.2}$$

In Eq.(1.2) c_k and r_j are the penalisation coefficients for inequality and equality constraints respectively. The quantities $G_k(\mathbf{x})$ and $H_j(\mathbf{x})$ are defined as:

$$\begin{aligned} G_k(\mathbf{x}) &= \max[0, g_k(\mathbf{x})] \qquad k = 1, ..., r \\ H_j(\mathbf{x}) &= \max[0, \mid h_j(\mathbf{x}) \mid - \epsilon] \quad j = 1, ..., m \end{aligned} \tag{1.3}$$

In Eq.(1.2) and (1.3) the equality constraints have been transformed into inequality constraints having the form $\mid h_j(\mathbf{x}) \mid \leq \epsilon$. Concerning the parameters c_k and r_j, in classical penalisation methods, the user must set their values to an appropriate level in order to ensure the search of solutions for the optimisation problem to be forced within the feasible domain. Nevertheless, the choice of these coefficients is very difficult and it is common practice to estimate their values by trial and error. Moreover, it could be useful to adjust penalisation pressure along the generations by tuning these coefficients, but this is directly linked on a guess or on a deep knowledge of the nature of the optimisation problem by the user.

The idea of the ADP is that it is possible to exploit the information restrained in the population, at the current generation, in order to guide the search in the case of constrained optimisation problem. Generally, in the first generation the population is generated randomly. With high probability the individuals are evenly distributed over both feasible and unfeasible domain and the corresponding values of objective functions and constraints can be used to estimate an appropriate level of penalisation, i.e. the values of penalisation coefficients c_k and r_j. At the current generation, inside the population it is possible to separate feasible and unfeasible individuals and

it is also possible classify each group in terms of increasing values of the objective function or constraint violation. The first individual in each group is the best candidate to be solution of the optimisation problem on the feasible and unfeasible side of the domain, respectively. One possible definition of the penalisation coefficients is the follow:

$$\begin{aligned} c_k(t) &= \frac{\mid {f_{best}}^F - {f_{best}}^{NF} \mid}{{G_{k\,best}}^{NF}} \quad k = 1, ..., r \\ r_j(t) &= \frac{\mid {f_{best}}^F - {f_{best}}^{NF} \mid}{{H_{j\,best}}^{NF}} \quad j = 1, ..., m \end{aligned} \tag{1.4}$$

In Eq.(1.4) the coefficients c_k and r_j are evaluated at the current generation t, while the apexes F and NF stand for feasible and non-feasible respectively. It is clear that the estimation of penalisation coefficients, according to the Eq.(1.4), can be repeated at each generation, thus tuning the appropriate penalisation pressure on the current population. The main advantages of this approach are substantially two: first of all this procedure is *automatic* because the GA can automatically calculate the values of the penalisation coefficients, secondly the method is *dynamic* since the evaluation of the penalisation level is updated at each generation.

1.3 General features of BIANCA

The version 3.1 of BIANCA shows several original features. As well as the previous version, one of the main features is the decomposition of the GA in a certain number of *macros*: it is possible to assembly them in various ways in order to suit many different optimisation problems and also to test and compare the effectiveness of different numerical strategies. In this sense, BIANCA is a bunch of genetic tools which the user can use as bricks to build up several GAs.

Another important feature of BIANCA is the representation of the information which is rich and detailed, but also non redundant. The biological metaphor in GAs is a simple but powerful mean to return the richness and completeness of information linked to design variables. The information restrained in the population along the generation is treated in a peculiar manner in such a way to allow a deep mixing of the individuals' genotype by means of the reproduction operators, i.e. cross-over and mutation, which act on every single gene of the individuals. For a deep insight the matter see [2].

In order to allow the reproduction phase among individuals belonging to different species, in BIANCA 3.1 the structure of the individual and, con-

sequently, the representation of the information as well as the reproduction operators of cross-over and mutation have been modified in order to deal with the optimisation problem of modular systems. We have introduced new genetic operators in BIANCA 3.1 for cross-over and mutation of individuals belonging to different species.

BIANCA 3.1 has the following qualities:

- objective function evaluation: a library of functions corresponding to objective and constraint functions of different optimisation problems, in addition in BIANCA the user can write its model by means of a special *macro*;
- fitness evaluation: several choices are available for fitness evaluation depending on the kind of problem, i.e. minimisation or maximisation, and on the selection pressure that the user decides to introduce. The fitness is evaluated in such a way that the fitness function can assume all the possible values in the range [0 1];
- selection: two known techniques of selection are included, i.e. roulette wheel, tournament;
- standard genetic operators: the main genetic operators are cross-over and mutation, applying with a certain probability on each gene of the individual's genotype;
- additional genetic operators: *elitism* operator which preserve the best individual during each generation;
- handling constraints: automatic dynamic penalisation method for handling constraints;
- handling multiple populations: the need to simultaneously explore different regions of the design space, as well as the search of optima responding to distinct design criteria, led us to introduce the option of working with multiple populations in BIANCA. Moreover, a *migration* operator has been introduced in order to allow exchanges of informations between populations evolving through parallel generations. This *migration* operator is the classical *ring-type*.
- stop criterion: maximum number of generations reached or test of convergence, i.e. no improvements of the mean fitness of the population after a given number of cycles.

- new genetic operators: to deal with the problem which considers the number of variables among the optimisation variables, as in the case of modular systems, new genetic operators have been developed, such as the *chromosome shift* operator, the *chromosome reorder* operator, the *chromosome addition/deletion* operator. These new operators modify the reproduction phase allowing the reproduction among individuals of different species, see [3];

- interface with external software: if the model to be optimised is written in an external environment, it is possible to call it by BIANCA 3.1 in order to evaluate the objective function and constraints, and then to pass the design variables to the model. The logical scheme of the structure of the interface is shown in Fig. 1.2.

- post-processing of results: graphical results concerning the trend of the best feasible solution and the average value of the objective function along the generation are obtained through the creation of a .m file (by means of MATLAB® environment) that reads the output files of BIANCA 3.1.

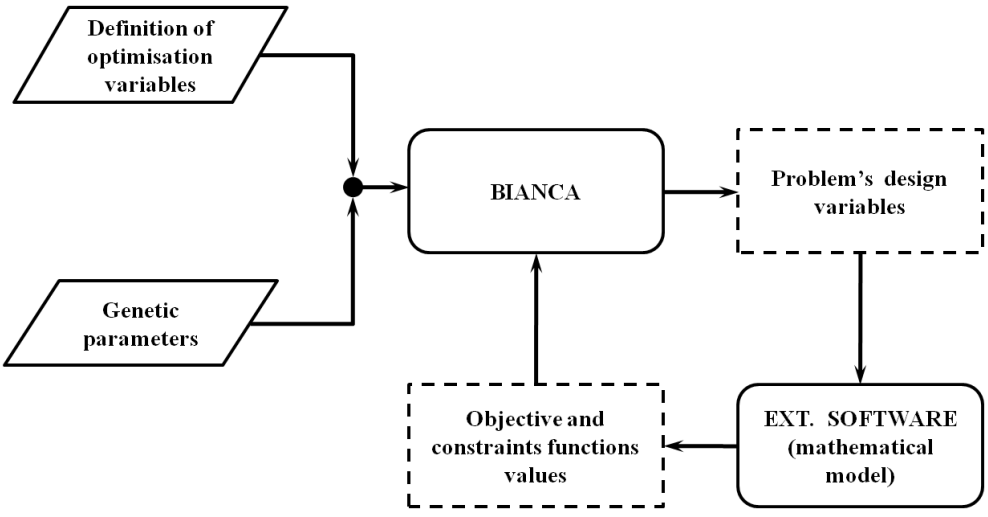

Figure 1.2: Logical scheme of the structure of the interface with external software in BIANCA.

1.4 The structure of the individual's genotype

The biological metaphor in GAs is a simple but powerful mean to return the richness and completeness of information linked to design variables. The necessity to deal with any type of design variables, i.e. continuous, discrete, scattered led us to the choice of a discrete representation of the information. As explained in Sec. 1.5, it exits a two-way relation among the variables and the *pointers*, i.e. integer numbers, which refer to the set of feasible discrete values for each variable. In a standard GA it is usual to encode integer values in the form of binary strings, in order to use the minimalist alphabet which increases the number of *schemes*, according to the theorem of the *implicit parallelism* that ensures improvement of the exploration of the domain and exploitation of information [5, 6]. In addition the use of binary representation allows the use of binary cross-over and mutation which are very effective when dealing with particular classes of optimisation problems.

In the previous version of BIANCA, an individual was represented by an array of dimensions $n_{chrom} \times n_{gene}$. The number of rows, n_{chrom}, is the number of chromosomes, while the number of columns, n_{gene}, is the number of genes. Basically, each design variable is coded in the form of a gene, and its meaning is linked both to the position and to the value of the gene within the chromosome. In principle, no limit is imposed on the number of genes and chromosomes for an individual in BIANCA. A number n_{ind} of individuals compose a population, and in BIANCA it is possible to work, at the same time, with several populations whose number, n_{pop}, can be defined by the user.

In order to include the number of chromosomes (i.e. of design variables) among the design variables, and then to allow the reproduction among individuals belonging to different species, some modifications of the individual genotype have been done. In BIANCA 3.1, the genotype of each individual is represented by a binary array shown in Fig. 1.3. In this picture, the quantity $(g_{ij})^k$ represents the j^{th} gene of the i^{th} chromosome of the k^{th} individual. Letter e stands for empty location, i.e. there is no gene in this location while n^k is the k^{th} individual's chromosomes number. It appears clearly that every individual can have a different number of chromosomes, i.e. each individual can belong to a different species.
As an example, for a composite laminate, one can assume, as design variables, the layers number, orientation angles and thickness. The information structure (i.e., in the GA's language, the *genotype*) of the individual-laminate is then structured as shown in Fig. 1.4. In this case the k^{th} laminate's number

of layers is n^k while the orientation and the thickness of the i^{th} ply are δ_i and h_i, respectively. One can notice that the number of layers n^k is the number of chromosomes of the k^{th} individual, while the orientation and thickness of the i^{th} layer are the two genes of the i^{th} chromosome.

$(g_{11})^k$	$(g_{12})^k$	...	$(g_{1m})^k$	n^k
$(g_{21})^k$	$(g_{22})^k$	...	$(g_{2m})^k$	
...	...	...	...	
...	...	...	...	
$(g_{n1})^k$	$(g_{n2})^k$	...	$(g_{nm})^k$	
e	e	e	e	

Figure 1.3: Structure of the individual's genotype with variable number of chromosome in BIANCA 3.1.

$(\delta_1)^k$	$(h_1)^k$	n^k
$(\delta_2)^k$	$(h_2)^k$	
...	...	
...	...	
$(\delta_n)^k$	$(h_n)^k$	
e	e	

Figure 1.4: Example of individual-laminate with variable number of layers in BIANCA.

1.5 Encoding/decoding of the variables

In BIANCA, the representation of the definition domain, i.e. the *design space*, of each design variable is made by the use of *pointers*, which are

themselves integer values. It exists a two-way relation between the variables and the *pointers*. This relation is clear in the case of discrete or grouped variables, in fact if the domain of definition have a finite dimension N, it is possible to enumerate all admissible values $v_i, (i = 1, ..., N)$ and build a reference between each value v_i and its index i, i.e. the *pointer* of that value. When the definition domain does not have finite dimension, it is necessary to restrict it, defining lower and upper bounds to the space of admissible values of v_i, i.e. v_{min} and v_{max} respectively. In the case of continuous variables, the first step is the discretisation of the definition domain by choosing a given precision p, and then it is possible to apply the same system of referencing by *pointers* as for discrete and grouped variables, see Fig.1.5.

In BIANCA *pointers* constitute the genotype of the individual, more precisely the single *pointer* corresponds to a gene, and all genetic operators are directly applied on the *pointers* representing the variables. Therefore, a step of decoding/encoding is necessary to translate the value of the pointer into the corresponding value of the design variable, and vice-versa. More details can be found in [1, 2].

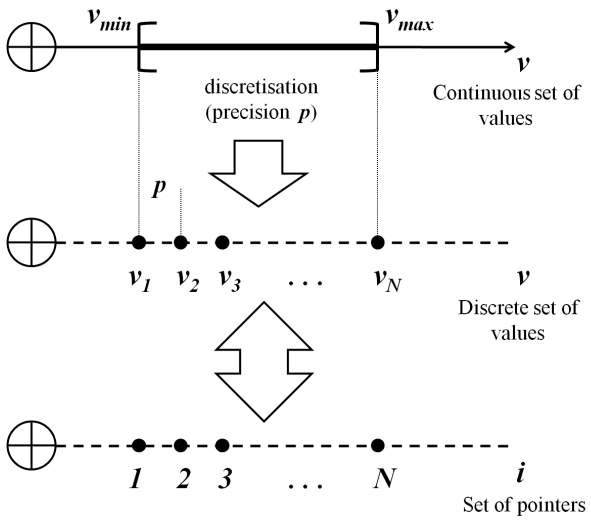

Figure 1.5: Two-way relation between continuous variables and *pointers* in BIANCA.

Chapter 2

BIANCA tutorial

2.1 Compiling BIANCA

The BIANCA batch file is named BIANCA 3.1.bat. You can compile BIANCA by a simple double click on this file. In the BIANCA 3.1 folder you must have the following files:

- *libBIANCA.a*: this is a library of BIANCA *macros* containing all the subroutines that BIANCA needs to run;
- MACRO_MY_PROBLEM.f95: this is the subroutine that you must use if you want to realise your model in FORTRAN environment as subroutine of BIANCA. The structure of this *macro* is explained in Sec 2.5.

After the compilation, the executable file BIANCAv3.1.exe is created.

2.2 Running BIANCA

The BIANCA executable file is named BIANCAv3.1.exe. You can run the code in two different way:

- by double click on the icon BIANCAv3.1.exe;
- by entering the command BIANCAv3.1 in the command prompt, after you have specified the correct path for BIANCAv3.1.

In both case, after the run of BIANCA, the code requires the specification of the *name of the current job session*. The choice of the job's name is completely arbitrary, but it must observe the following condition: *the name of the current job session must be the same as the two input files with extension .gen and .opt*, described in the following section.

2.3 Inputs to BIANCA

There are two different kind of inputs for BIANCA. In particular, the main inputs of the code are written in two input files with extension *.gen* and *.opt* respectively. As explained beforehand, these files must have the same name as the current job session.
The input file with extension *.gen* contains the genetic parameters of the simulation, whilst the one with extension *.opt* contains the optimisation parameters.

Moreover, in BIANCA there are two additional input files that have fixed name and structure, i.e. you can modify these input files but you can not change their name. These files are *library.inp* and *post_processing.inp* and they contain the information about some library functions concerning the laminates' design (already implemented within BIANCA) and the information about the post-processing operations, respectively.
In the following subsections the structure of all these input files is described in details.

2.3.1 Genetic parameters

As already explained, the input file with extension *.gen* contains the genetic parameters of the simulation. The structure of the file is defined below (we remark that each item-number in the list corresponds to the information restrained in a single line of the file):

1. n_{pop}, number of population (integer), the maximum allowable number of population is 10;

2. n_{ind}, number of individuals (integer), the maximum allowable number of individuals is 2000;

3. *stop crit.*, stop criterion (character):

   - *fixed_generations*, to stop the GA after a given number of the generation;
   - *threshold*, to stop the GA when the best individual satisfy the sill value on the objective function;
   - *mixed*, is a combination of the two previous criteria;

4. in this line the user must write some values according to the stop criterion selected in *line* 3:

- n_{gen}, number of generations (integer) if *fixed_generations*;
- $tresh$, threshold value (double precision) if *threshold*;
- $tresh \quad n_{gen}$, threshold value (double precision) and number of generations (integer) if *mixed*;

5. p_{cross}, crossover probability (double precision);
6. p_{mut}, mutation probability (double precision);
7. p_{shift}, shift operator probability (double precision);
8. $p_{mutchrom}$, mutation probability of the number of chromosomes (double precision);
9. SEL, selection operator (integer):
   - 1, for roulette wheel selection;
   - 2, for tournament selection;
10. $fit.\ pres.$, fitness pressure (double precision);
11. $ELIT$, elitism strategy (integer):
    - 0, the elitism is not applied;
    - 1, the elitism is applied;
12. I_{time}, isolation time (integer): when in *line* 1 n_{pop} is greater then 1 you must choice this value. It represents the number of generation during which the populations are isolated. Every I_{time} generations an exchange of the best feasible individuals among the populations is realised.

As example, we show here the structure of the *.gen* input file.

```
1
100
fixed_generations
200
0.85
0.01
0.5
0.04
1
1.0
```

```
1
0
```

In this example we use a single population with 100 individuals. The stop criterion is based on a fixed number of generations, i.e. the code stops the simulation after 200 generations; crossover probability is equal to 0.85, mutation probability is equal to 0.01, shift operator probability is 0.5 whilst the mutation probability of the number of chromosomes is 0.04. The selection operator is roulette wheel selection and it acts with a fitness pressure of 1.0. The elitism strategy is applied and the isolation time is set equal to 0 because in this example we have only one population.

2.3.2 Optimisation parameters

As said previously, the input file with extension *.opt* contains the optimisation parameters of the simulation. The structure of the file is defined below (we remark that each item-number in the list corresponds to the information restrained in a single line of the file):

1. ENV, flag variable (character) that denotes the type of the environment in which your physical model is realised:
   - *internal*, if the model is realised in FORTRAN language as subroutine of BIANCA or if you want to use some internal function implemented within BIANCA ;
   - *external*, for models realised in a different environment by means of external codes;
2. $KINDF$, flag variable (character) that must be set only if you want to perform an optimisation with some functions already written within BIANCA or if you want to write your physical model in FORTRAN language (when in *line* 1 the *internal* option is active):
   - *test_function*, to access to the library of BIANCA test functions;
   - *library_function*, to access to BIANCA composite laminate function;
   - *my_problem*, if you want to write your physical/mathematical model in FORTRAN environment. The model must be written as subroutine of BIANCA;

3. *CODE*, flag variable (character) which must be set only if you want to perform an optimisation process on a model realised by means of external software (when in *line* 1 the *external* option is active):

   - *MATLAB*, for models realised in MATLAB® environment;
   - *ANSYS*, for models realised in ANSYS® environment;
   - *ABAQUS*, for models realised in ABAQUS® environment;

4. *MODEL NAME*, name of the file (character) that describe the physical/mathematical model which you want to optimise (valid only when in *line* 1 the *external* option is active). **BEWARE**: the name of the file *must contain the extension* (e.g. for an ANSYS file a possible name can be *cantilevered_beam.lgw*). **EXCEPTION**: in case of MATLAB files the user do not write the extension (e.g. not *rotorcraft_dynamic.m* but *rotorcraft_dynamic*);

5. *MODEL I*, name of the input file (character) which passes the design variables from BIANCA to the external model (valid only when in *line* 1 the *external* option is active). **BEWARE**: the name of the file must contain the extension. The structure of this file is explained in Sec. 2.6;

6. *MODEL O*, name of the output file (character) which passes the values of constraint and objective functions from the external model to BIANCA (valid only when in *line* 1 the *external* option is active). **BEWARE**: the name of the file must contain the extension. The structure of this file is explained in Sec. 2.6;

7. *IDF*, ID of the internal function written inside BIANCA (integer) (valid only when in *line 1* the *internal* option is active):

   - if in *line* 2 the *test_function* option is active:
     - 6, Vannucci's function with one inequality constraint;
     - 60, Vannucci's function without constraints;
     - 61, Vannucci's function with one equality constraint;
     - 80, Welded beam design problem, see [];
     - 81, Pressure vessel design problem, see [];
     - 82, Tension compression spring weight design problem, see [];
   - if in *line* 2 the *library_function* option is active:

  - 71, model and functions for the optimisation of the laminates' elastic symmetries with fixed number of plies, i.e. chromosomes (**BEWARE**: if this option is active you must choose one or many objective functions from the *library.inp* file);
  - 72, model and functions for the optimisation of the laminates' elastic symmetries with variable number of plies, i.e. chromosomes (**BEWARE**: if this option is active you must choose one or many objective functions from the *library.inp* file);

8. $MAXORMIN$, ID for maximisation or minimisation problems (integer):

   - 1, for maximisation;
   - 2, for minimisation;

9. n_{obj}, number of partial objective functions (integer). If your optimisation problem presents an objective function which is constituted by the sum of different terms, you can use this options in order to see the evolution of the different partial objectives along the generations. **BEWARE**: this options must be used when you want to perform an optimisation problem with the internal library function for the laminates' design of BIANCA, i.e. when in *line* 7 the function ID is 71 or 72. In this case the total objective function is composed by the sum of different terms and every one is linked to a particular elastic symmetry of the laminate. For more details see [1, 2, 3, 7]. The maximum allowable number of partial objective functions is 30;

10. $IDCONSTR$, ID for constraints (integer):

    - 0, if constrains are inactive;
    - 1, if constrains are active;

11. n_{constr}, number of constraints (integer), the maximum allowable number of constraint functions is 30;

12. $CHROMVAR$, optimisation with individuals with variable number of chromosomes (character):

    - *yes*, for individuals having different number of chromosomes;
    - *no*, otherwise;

13. *CHROMMIN exp*, optimisation with minimum number of chromosome (character) and exponent of minimum chromosome function (double precision). **BEWARE**: this option must be set only when in *line* 2 the *library_function* option is active and when in *line* 12 the *yes* option is active:

    - *yes p*, if you want to perform the optimal laminates' design with the minimum number of plies and with an exponent p on the chromosome function;
    - *no 0.0*, otherwise;

14. $n_{chrommin}$ $n_{chrommax}$, minimum and maximum number of chromosomes (integers),(valid only when in *line* 12 the *yes* option is active), the maximum allowable number of chromosomes is 50;

15. n_{chrom}, number of chromosomes (integer), (valid only when in *line* 12 the *no* option is active) the maximum allowable number of chromosomes is 50;

16. n_{gene}, number of genes (integer), the maximum allowable number of genes is 50;

17. n_{var}, number of different type of variables (integer). **BEWARE**: the number of different type of variables can be different from the number of genes, e.g. in a particular optimisation problem we can have 3 design variables but only 2 different types of variables. We show below an example about this particular case;

18. *BLANK LINE*

FOLLOWING LINES: the lines that follow the *line* number 18 must be occupied by the declaration of variables. Depending on the kind of variables we can have 4 or 5 *lines* for each variable. **BEWARE**: *After the description of each variable you must put a BLANK LINE.* Table 2.1 describes the different nature of the variables and the number of lines that the declaration occupies.

LAST LINE: $chromo - mask$, chromo-mask (vector of integers). This mask denotes the position of the genes within the chromosomes of all individuals. This mask also shows the nature of the variable linked to each gene. This line is then constituted by a sequence of integer numbers, e.g. if in a given optimisation problem there are two different type of variables and two genes this line must be written in the following way: *1 2*.

Continuous variables	**Regular discrete variables**	**Scattered discrete variables**
Variable name (character)	Variable name (character)	Variable name (character)
EXTENDED	*REGULAR_DISCR*	*SCATTERED_DISCR*
Left bound value (double precision)	Left bound value (double precision)	Vector of scattered values (double precision)
Right bound value (double precision)	Right bound value (double precision)	
	Discretization step (double precision)	
4 *lines*	5 *lines*	3 *lines*

Table 2.1: Declaration of different types of variables in BIANCA

As example, we show here the structure of the *.opt* input file.

```
internal
test_function
none
none
none
none
6
2
1
1
1
no
no 0.0
0 0
1
2
2

x_1
EXTENDED
0.0
```

```
12.566

x_2
EXTENDED
0.0
6.283

1 2
```

For this example we use an internal test function written within BIANCA: the Vannucci's function with one inequality constraints. We want to minimise this function. So we perform an optimisation process with fixed number of chromosome. The structure of the individual's genotype is made by a single chromosome with two genes and, hence, with two different type of variables. The name of the first variable is *x_1* while the name of the second one is *x_2*. Both variables are continuous, and we have defined their bounds. *x_1* varies in a continuous way between 0.0 and 4π, while *x_2* varies in a continuous way between 0.0 and 2π. Since there are two different types of variables the chromo-mask is made by two components 1 2.

Concerning the meaning of the parameter n_{var}, i.e. the number of different types of variables, and difference between this parameter and the number of genes n_{gene} we show here below an example in order to understand in a better way how you must compile the *.opt* input file for this kind of situation:

```
internal
my_problem
none
none
none
none
0
2
1
1
4
no
no 0.0
0 0
2
3
2
```

```
angle
EXTENDED
0.0
90.0

height
REGULAR_DISCR
2.0
5.0
0.1

1 2 1
```

In this example we use an user self-created model, written in FORTRAN language as subroutine of BIANCA. The model has 1 objective function with 4 constraints. We want to minimise this function. The structure of the individual's genotype is made by 2 chromosomes with 3 genes, hence we have 6 design variables but only 2 different types of variables. The name of the first variable's type is *angle* while the name of the second one is *height*. The first type is continuous, whilst the second one is a discrete variable. We have also defined their bounds. *angle* varies in a continuous way between 0.0° and 90°, while *height* varies between 2.0mm and 5.0mm, with a discretisation step of 0.1mm. Since there are 2 different types of variables but 3 genes the chromo-mask is made up by 3 components: 1 2 1. The chromo-mask identifies the position of each design variable and also of each gene within the chromosome. So in this example each one of the two chromosomes, that constitute the genotype of the individual, is composed by 3 variables belonging to 2 different types: the first variable associated to the corresponding gene is an angle, e.g. α, and it varies between the bounds described by the variable type *angle*, the second variable is a height h and varies between the bounds described by the variable type *height*, while the third variable is also an angle, e.g. ϕ, which has a different physical meaning in our problem but that belongs to the same type of variable as the first one.

2.3.3 The *library.inp* file

The *library.inp* file must be compiled only when in the *line* 1 and *line* 2 of the *.opt* input file the options *internal* and *library_function* are active, respectively. Moreover, in the *line* 7 of the *.opt* input file the only valid ID are 71 or 72.

The ID 71 identifies the problem of the optimal design of laminates' elastic symmetries with fixed number of plies, i.e. chromosomes: in this case the only design variables are the plies' orientations. The user must declare the bounds, and eventually the discretisation step, in degrees. A little remark occurs for this kind of design problem: the number of plies is equal to the number of chromosomes increased by one, e.g. when you set, in the *.opt* input file, the number of chromosomes equal to 10 you perform an analysis on a laminate with 11 layers, where the first layer has a fixed orientation equal to 0°. An example of this kind of analysis can be found in Sec. 3.2.1. The ID 72 identifies the problem of the optimal design of laminates' elastic symmetries with variable number of plies, i.e. chromosomes: in this case you must perform an analysis with cross-over on species and the design variables are the plies' orientations and thickness. The user must declare the bounds, and eventually the discretisation step, of the orientations in degrees, while the ones of the thickness must be declared in mm. A little remark occurs for this kind of design problem: unlike the previous function ID, in this case the number of plies is equal to the number of chromosomes. An example of this kind of analysis can be found in Sec. 3.2.2.

The *library.inp* file presents a list of all the possible elastic symmetries of the laminate in terms of stiffness and compliance matrices. Each symmetry is characterised by a name, and every names are written in the file itself. You can copy the name of each symmetry over the *line* which contains the following words: *Don't modify the following lines.* You can see an example of the structure of this file in Fig. 2.1.

```
homogeneity
uncoupling


------ Don't modify the following lines ------------------------------------------------------------

Choose the objective functions within the following list and copy one of them over the previous line.

------------ DECOUPLING ----------------------------------
1)uncoupling

------------ HOMOGENEITY ---------------------------------
2)homogeneity

------------ MEMBRANE STIFFNESS --------------------------
3)membrane_stiffness_R0_orthotropy
4)membrane_stiffness_R1_orthotropy
5)membrane_stiffness_orthotropy_K=0
6)membrane_stiffness_orthotropy_K=1
7)membrane_stiffness_isotropy

------------ BENDING STIFFNESS ---------------------------
8)bending_stiffness_R0_orthotropy
9)bending_stiffness_R1_orthotropy
10)bending_stiffness_orthotropy_K=0
11)bending_stiffness_orthotropy_K=1
12)bending_stiffness_isotropy

------------ MEMBRANE COMPLIANCE -------------------------
13)membrane_compliance_r0_orthotropy


------------ BENDING COMPLIANCE --------------------------
14)bending_compliance_r0_orthotropy


------------ AXIS COINCIDENCE-STIFFNESS ------------------
15)R0_stiffness_axis
16)R1_stiffness_axis
17)orthotropy_stiffness_axis
18)R0m_orthob_stiffness_axis
19)R0b_orthom_stiffness_axis


------------ AXIS COINCIDENCE-COMPLIANCE -----------------
20)r0_compliance_axis


------------ AXIS COINC COMPLIANCE-STIFFNESS -------------
21)r0b_compliance_R0m_stiffness_axis
22)r0b_compliance_orthom_stiffness_axis
23)r0m_compliance_R0b_stiffness_axis
24)r0m_compliance_orthob_stiffness_axis
```

Figure 2.1: Structure of the *library.inp* input file.

BEWARE: the *number* of elastic symmetries must be equal to the number of partial objective function written in the input file with extension *.opt* at *line* 9.
We show here below an example of the structure of both *.opt* and *library.inp* input files.

.opt input file

```
internal
library_function
none
none
none
none
71
2
```

```
3
0
0
no
no 0.0
0 0
12
1
1

angle
REGULAR_DISCR
-90.0
90.0
5.0

1
```

library.inp input file

```
membrane_stiffness_isotropy
bending_stiffness_orthotropy_K=0
uncoupling
```

In this example we want to find a laminate which posses the following elastic symmetries:

- elastic uncoupling;
- in-plane stiffness isotropy;
- bending stiffness orthotropy.

We use the internal function 71 and the objective function is made up by 3 partial objective: each one is associated to the symmetries cited beforehand. The number of chromosomes is associated to the number of plies (**BEWARE**, for the library function 71 the number of laminates'plies is equal to the number of chromosomes increased by one, so in this example we have 12 chromosomes and the laminate has 13 plies. In fact the first ply is

oriented at 0.0°). For each chromosome-ply we have one gene, i.e. one design variable: the angle which varies with a discretisation step of 5.0° between −90.0° and 90.0°. We remark that the number of partial objective function is 3 and is equal to the number of the elastic symmetries written in the file *library.inp*.

2.3.4 The *post_processing.inp* file

The *post_processing.inp* input file contains several options that you can set if you want to perform the graphical post processing of the simulation's results.

Post processing operations are performed via MATLAB® software. At the end of the simulation BIANCA writes a file named *graphic_results.m* which contains all the instructions for the plotting of the results. This file automatically reads the data restrained in the output files of BIANCA, whose structure is explained in the next Section.
These MATLAB® instructions are strictly linked with what you write in the *post_processing.inp* file. The graphical results that you obtain, after you have compiled in a correct way the *post_processing.inp* input file, are substantially 2 files whose name is the same as the current job session preceded by the following words:

- *obj_min_(job's session name)* for the file which contains the plot of the best feasible solution vs. generations;
- *obj_mean_(job's session name)* for the file which contains the plot of the average value of the penalised objective function vs. generations;

The structure of the *post_processing.inp* input file is defined below (we remark that each item-number in the list corresponds to the information restrained in a single line of the file):

1. *POST ENV*, post processing environment (character). This is a flag variable that you must set if you want to activate the graphical treatment of results:
   - *MATLAB*, if you want to use MATLAB® package for the post processing of results;
   - *none*, otherwise;
2. *axis*, flag variable (character) that allows you to choice the axis kind of the results' plot:
   - *linear*, for a linear scale;

- *semilog*, for a semi-logarithmic scale;

3. *grid*, flag variable (character) that allows you to activate/deactivate the grid on the plots:

   - *on*, grid is active;
   - off, grid is not active;

4. *plot format*, flag variable (character) that allows you to choice the format for the simulation's plots:

   - *bmp*, the graphical results are saved as Windows bitmap file;
   - emf, the graphical results are saved as Enhanced metafile;
   - *eps*, the graphical results are saved as EPS level 1;
   - *jpg*, the graphical results are saved as JPEG image;
   - *pbm*, the graphical results are saved as Portable bitmap;
   - *pcx*, the graphical results are saved as Paintbrush 24-bit;
   - *pdf*, the graphical results are saved as Portable Document Format;
   - *pgm*, the graphical results are saved as Portable Graymap;
   - *png*, the graphical results are saved as Portable Network Graphics;
   - *ppm*, the graphical results are saved as Portable Pixmap;
   - tif, the graphical results are saved as TIFF image, compressed;

We show here below an example of the structure of *post_processing.inp* input file.

```
MATLAB
linear
off
png
```

In this case we have performed a post processing treatment of results via MATLAB software. We obtain two file *obj_min_(job's session name).png* and *obj_mean_(job's session name).png* which contain the results of the simulation. The graphics are in linear scale and without grid.

2.4 Outputs from BIANCA

In BIANCA, at the end of the optimisation process, we have 3 output files. The name of these files is the same as the one of the current job session. These output files have 3 different extension: *.bio*, *.pop* and *.sta* respectively. We describe the structure and the contents of those files in the following subsections.

2.4.1 The *.bio* output file

This file contains the informations about the *best feasible individual* for every generations. In particular in this file we can find:

- the number of generations;
- the ID of the population;
- the ID of the best individual within the population;
- the name and the number of each design variable for each chromosome of the best individual;
- the value of the non-penalised partial objective functions of the best individual;
- the value of the constraint functions of the best individual;
- the value of the total objective function of the best individual.

Fig. 2.2 shows the structure of the *.bio* output file for the Vannucci's function problem with one inequality constraint. In this simulation we consider a population of 10 individuals evolving through 50 generations.

GENER	NPOP	NIND	x_1 1	x_2 1	obj 1	ctr_in 1	OBJ_TOT
0	1	3	0.964253E+01	0.219260E+01	-.136214E+01	-.140772E+00	-.136214E+01
1	1	3	0.762804E+01	0.469843E+01	-.467262E+01	-.368821E+01	-.467262E+01
2	1	1	0.101830E+02	0.307087E+01	-.493883E+01	-.600285E+00	-.493883E+01
3	1	9	0.816851E+01	0.469843E+01	-.500306E+01	-.347134E+01	-.500306E+01
4	1	2	0.107726E+02	0.312615E+01	-.753542E+01	-.100884E+00	-.753542E+01
5	1	2	0.107726E+02	0.312615E+01	-.753542E+01	-.100884E+00	-.753542E+01
6	1	1	0.107726E+02	0.307701E+01	-.781126E+01	-.517496E-01	-.781126E+01
7	1	7	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
8	1	5	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
9	1	2	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	2	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
11	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
12	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
13	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
14	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
15	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
16	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
17	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
18	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
19	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
21	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
22	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
23	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
24	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
25	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
26	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
27	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
28	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
29	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
31	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
32	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
33	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
34	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
35	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
36	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
37	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
38	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
39	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
41	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
42	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
43	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
44	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
45	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
46	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
47	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
48	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
49	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01

Figure 2.2: Structure of the *.bio* output file.

2.4.2 The *.pop* output file

In the output file with extension *.pop* the informations about the whole population are written every 10 generations. Concerning the informations about the whole population we can find:

- the number of generations;
- the ID of the population;
- the ID of every individual within each population;
- the name and the number of each design variable for each chromosome for each individual;
- the value of the non-penalised partial objective functions for each individual;
- the value of the constraint functions for each individual;

- the value of the total objective function for each individual.

Fig. 2.3 shows the structure of the *.pop* output file for the Vannucci's function problem with one inequality constraint. In this simulation we consider a population of 10 individuals evolving through 50 generations.

GENER	NPOP	NIND	x_1 1	x_2 1	obj 1	ctr_in 1	OBJ_TOT
0	1	1	0.487654E+01	0.399213E+00	0.232940E+01	-.689618E-01	0.232940E+01
0	1	2	0.117921E+02	0.303402E+01	-.700909E+01	0.127101E+01	0.308695E+01
0	1	3	0.964253E+01	0.219260E+01	-.136214E+01	-.140772E+00	-.136214E+01
0	1	4	0.762804E+01	0.388158E+01	0.294007E+00	-.287136E+01	0.294007E+00
0	1	5	0.116693E+02	0.503623E+01	0.964449E+01	-.911524E+00	0.964449E+01
0	1	6	0.816851E+01	0.155386E+01	0.145285E+01	-.326771E+00	0.145285E+01
0	1	7	0.123695E+02	0.988820E+00	0.877219E+00	0.428286E+01	0.348974E+02
0	1	8	0.100110E+02	0.356221E+00	0.373309E+01	0.197271E+01	0.194030E+02
0	1	9	0.192851E+01	0.124063E+01	-.121425E+00	-.119499E+01	-.121425E+00
0	1	10	0.966710E+01	0.532489E+01	0.216913E+01	-.325564E+01	0.216913E+01
10	1	1	0.107235E+02	0.302788E+01	-.788440E+01	-.533109E-01	-.788440E+01
10	1	2	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	3	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	4	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	5	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	6	0.107603E+02	0.302788E+01	-.801677E+01	-.153716E-01	-.801677E+01
10	1	7	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	8	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	9	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
10	1	10	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	2	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	3	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	4	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	5	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	6	0.107603E+02	0.302788E+01	-.801677E+01	-.153716E-01	-.801677E+01
20	1	7	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	8	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	9	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
20	1	10	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	2	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	3	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	4	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	5	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	6	0.107603E+02	0.302788E+01	-.801677E+01	-.153716E-01	-.801677E+01
30	1	7	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	8	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	9	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
30	1	10	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	2	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	3	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	4	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	5	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	6	0.107603E+02	0.302788E+01	-.801677E+01	-.153716E-01	-.801677E+01
40	1	7	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	8	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	9	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
40	1	10	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	1	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	2	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	3	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	4	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	5	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	6	0.107603E+02	0.302788E+01	-.801677E+01	-.153716E-01	-.801677E+01
50	1	7	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	8	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	9	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01
50	1	10	0.107726E+02	0.302788E+01	-.805880E+01	-.261572E-02	-.805880E+01

Figure 2.3: Structure of the *.pop* output file.

2.4.3 The *.sta* output file

This file contains the informations about the statistics on the whole population for each generation. In particular in this file we can find:

- the number of generations;
- the ID of the population;
- the number of each non-penalised partial objective function and the minimum and average values within the population for each generation;
- the minimum and average values of the total objective function within the population for each generation;
- the maximum and average values of the fitness function within the population for each generation;

Fig. 2.4 shows the structure of the *.sta* output file for the Vannucci's function problem with one inequality constraint. In this simulation we consider a population of 10 individuals evolving through 50 generations.

GENER	NPOP	N_OBJ	OBJ_MIN	OBJ_MEAN	OBJ_TOT_MIN	OBJ_TOT_MEAN	FIT_MAX	FIT_MEAN
0	1	1	-.700909E+01	0.120075E+01	-.136214E+01	0.717936E+01	0.100000E+01	0.764434E+00
1	1	1	-.467262E+01	-.277495E+00	-.467262E+01	0.568850E+00	0.100000E+01	0.583901E+00
2	1	1	-.493883E+01	-.203318E+01	-.493883E+01	-.290409E+00	0.100000E+01	0.774347E+00
3	1	1	-.500306E+01	-.291055E+01	-.500306E+01	-.273842E+01	0.100000E+01	0.544696E+00
4	1	1	-.753542E+01	-.415070E+01	-.753542E+01	-.392911E+01	0.100000E+01	0.412268E+00
5	1	1	-.753542E+01	-.539740E+01	-.753542E+01	-.539740E+01	0.100000E+01	0.548149E+00
6	1	1	-.781126E+01	-.524257E+01	-.781126E+01	-.519398E+01	0.100000E+01	0.529556E+00
7	1	1	-.805880E+01	-.721945E+01	-.805880E+01	-.721945E+01	0.100000E+01	0.752125E+00
8	1	1	-.805880E+01	-.756936E+01	-.805880E+01	-.756936E+01	0.100000E+01	0.741772E+00
9	1	1	-.805880E+01	-.781725E+01	-.805880E+01	-.781725E+01	0.100000E+01	0.538474E+00
10	1	1	-.805880E+01	-.803716E+01	-.805880E+01	-.803716E+01	0.100000E+01	0.875898E+00
11	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
12	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
13	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
14	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
15	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
16	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
17	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
18	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
19	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
20	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
21	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
22	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
23	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
24	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
25	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
26	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
27	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
28	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
29	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
30	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
31	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
32	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
33	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
34	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
35	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
36	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
37	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
38	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
39	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
40	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
41	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
42	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
43	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
44	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
45	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
46	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
47	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
48	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
49	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00
50	1	1	-.805880E+01	-.805460E+01	-.805880E+01	-.805460E+01	0.100000E+01	0.900000E+00

Figure 2.4: Structure of the *.sta* output file.

2.5 The *macro* MACRO_MY_PROBLEM.f95

If you want to perform an optimisation process on your model, by means of the GA BIANCA, one possible way to do that is to write your model in FORTRAN environment. In this case you must observe the following conditions:

1. you must write your physical/mathematical model in FORTRAN language;

2. you must write your model as subroutine of BIANCA.

The *macro* MACRO_MY_PROBLEM.f95 has been realised in order to allow you to write in an easily way your model in FORTRAN language and to understand how your model can be interfaced within the code BIANCA. This *macro* contains 2 subroutines named *my_problem* and *my_problem_var* which have some input and output quantities. Fig. 2.5 shows the structure of the *macro*.

```
!     ****************************************************************
!                          MACRO MY PROBLEM
!     ****************************************************************

      subroutine my_problem(npop,nind,nchrom,ngene,x_being,            &
     &                      n_obj,obj,constr_active,                   &
     &                      n_constr,constr_ineq)


      implicit none

      integer npop,nind,nchrom,ngene
      integer n_obj,n_constr,constr_active
      double precision,dimension(10,2000,50,50) :: x_being
      double precision,dimension(10,2000,30) :: obj
      double precision,dimension(10,2000,30) :: constr_ineq



      end subroutine my_problem



      subroutine my_problem_var(npop,nind,nchrom_min,nchrom_max,ngene, &
     &                          x_being,n_obj,obj,constr_active,       &
     &                          n_constr,constr_ineq)



      implicit none

      integer npop,nind,nchrom_min,nchrom_max,ngene
      integer n_obj,n_constr,constr_active
      double precision,dimension(10,2000,50,50) :: x_being
      double precision,dimension(10,2000,30) :: obj
      double precision,dimension(10,2000,30) :: constr_ineq



      end subroutine my_problem_var

```

Figure 2.5: Structure of the *macro* *MACRO_MY_PROBLEM.f95*.

In the next subsections we explain in detail the structure of the two subroutines restrained in this *macro* and how and when you can use them. We show also an example on how you can write your model as subroutine of BIANCA, by means of the MACRO_MY_PROBLEM.f95, in Sec. 3.3.

2.5.1 The *my_problem* subroutine

The *my_problem* subroutine must be used when you want to write your mathematical model in FORTRAN environment, as subroutine of BIANCA, and when you have an optimisation problem where the number of design variables is fixed. In this case the cross-over between individuals belonging to different species is no longer required and you have to perform a standard genetic optimisation process.

The input quantities of this subroutine are:

- *npop*, number of populations (input derived from the input file *.gen*, *line* 1);
- *nind*, number of individuals for each population (input derived from the input file *.gen*, *line* 2);
- *nchrom*, number of chromosomes of each individual (input derived from the input file *.opt*, *line* 15);
- *ngene*, number of genes within each chromosome of the individual (input derived from the input file *.opt*, *line* 16);
- *n_obj*, number of partial objectives (input derived from the input file *.opt*, *line* 9);
- *n_constr*: number of constraints (input derived from the input file *.opt*, *line* 11);
- *x_being*: phenotype of the whole population, the real size of this 4-dimensional array is $x\_being(npop, nind, nchrom, ngene)$. This array contains the value of each design variable, linked to each gene, for the whole population.

The output quantities of this subroutine are:

- *obj*: 3-dimensional array which contains the values of the partial objective functions for every individuals of each population. The real size of this array is $obj(npop, nind, n\_obj)$;

- *constr_ineq*: 3-dimensional array which contains the values of the inequality constraint functions for every individuals of each population. The real size of this array is $constr\_ineq(npop, nind, n\_constr)$.

2.5.2 The *my_problem_var* subroutine

The *my_problem_var* subroutine must be used when you want to write your mathematical model in FORTRAN environment, as subroutine of BIANCA, and when you have an optimisation problem where the number of design variables is also a variable of the process, such as the case of the optimisation of modular systems where the number of modules and, hence, the number of variables is also a design variable for the problem. In this case the cross-over between individuals belonging to different species is required and you have to perform a non-standard genetic optimisation process.

The input quantities of this subroutine are:

- *npop*, number of populations (input derived from the input file *.gen*, *line* 1);
- *nind*, number of individuals for each population (input derived from the input file *.gen*, *line* 2);
- *nchrom_min*, minimum number of chromosomes (input derived from the input file *.opt*, *line* 14);
- *nchrom_max*, maximum number of chromosomes (input derived from the input file *.opt*, *line* 14);
- *ngene*, number of genes within each chromosome of the individual (input derived from the input file *.opt*, *line* 16);
- *n_obj*, number of partial objectives (input derived from the input file *.opt*, *line* 9);
- *n_constr*: number of constraints (input derived from the input file *.opt*, *line* 11);
- *x_being*: phenotype of the whole population, the size of this 4-dimensional array can change for each individual because each one can belong to a different specie and can have a different number of chromosome. In particular, the effective number of chromosomes of each individual is also restrained into the phenotype in a particular position of the array; if we consider the j^{th} individual of the i^{th} population the number of

chromosomes of this individual, $nchrom(i, j)$, is uniquely individuated by the following equality: $nchrom(i, j) = x\_being(i, j, 1, ngene+1)$. So for each individual the real size of the 4-dimensional array *x_being* is $x\_being(npop, nind, nchrom(npop, nind), ngene + 1)$. This array contains the value of each design variable, linked to each gene, for the whole population.

The output quantities of this subroutine are:

- *obj*: 3-dimensional array which contains the values of the partial objective functions for every individuals of each population. The real size of this array is $obj(npop, nind, n\_obj)$;
- *constr_ineq*: 3-dimensional array which contains the values of the inequality constraint functions for every individuals of each population. The real size of this array is $constr\_ineq(npop, nind, n\_constr)$.

2.6 Structure of the interface with external codes in BIANCA

In several problems, the value of the objective function and/or of the constraints, cannot be computed analytically, but it has to be evaluated using special numerical codes. Typically, this is the case of structural optimization, where the most part of times the structural response is numerically assessed using finite element (FE) codes. For these cases, a very general interface has been developed, which renders BIANCA able to exchange input/output informations with mathematical models supported by an external software.

Fig. 2.6 shows the structure of the data-exchange between BIANCA and a generic external software. For each individual, BIANCA performs the genetic operations, such as selection, cross-over, mutation and so on, and then passes the design variables to the mathematical model written in a different environment. At this point, the external software evaluates the objective and the eventual constraint functions values, and then passes them back to BIANCA. The data-exchange between BIANCA and the external software is simply done by means of two I/O files.

The first one is the file written from BIANCA and passed to the external software, i.e. the *input file*, which contains the informations related to the current individual at the current generation, i.e. the number and the values of the design variables restrained in that individual's genotype. This *input file* also contains additional information such as the number of objective and

constraint functions and also a particular variable, i.e. the *counter* whose meaning is explained in the next subsection.
The second one is the file written from external software and passed to BIANCA, i.e. the *output file*, in which are written the values of objective and constraint functions, and once again the value of the *counter*.

The writing operations of these files are made for every individual in the current generation, so the external code, during the whole optimisation process, is called from BIANCA $N_{ind} \times N_{gen}$ times, where N_{ind} is the number of individuals while N_{gen} is the number of generations.

Some current and well known software packages have been interfaced with BIANCA in this way, like for instance MATLAB®, CAST3M®, ABAQUS® and ANSYS®.

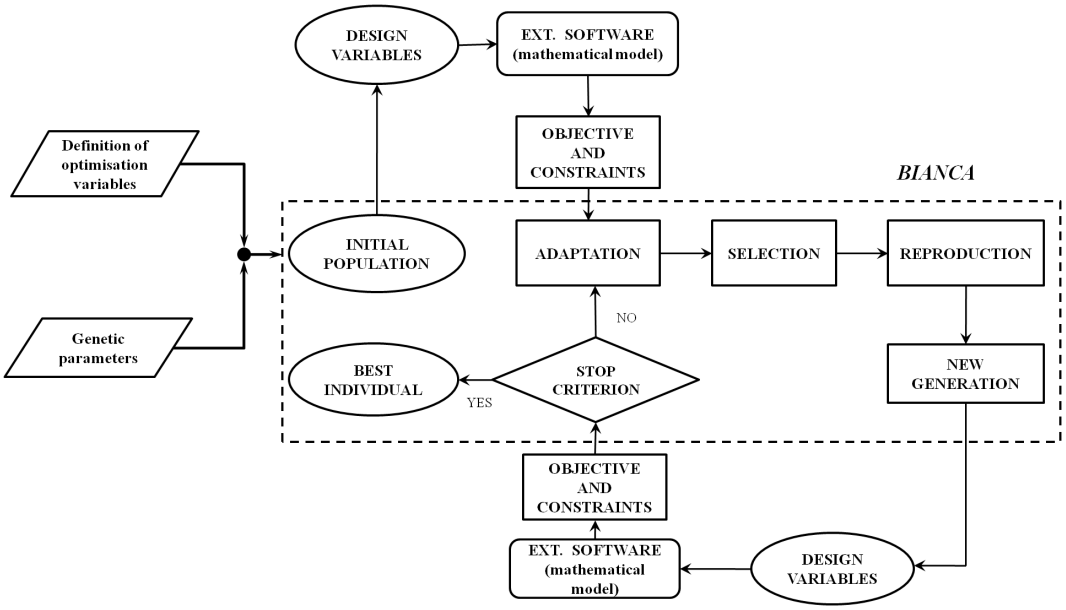

Figure 2.6: Structure of interface with external software in BIANCA.

2.6.1 The *input file* from BIANCA to the external code

As previously said, during the whole optimisation process, BIANCA passes an *input file* to the external code. You must read the value of design variables which your mathematical model (realised within your external code) needs from this file. The name of this file **must be the same** as the one you declare in *line* 5 of the *.opt* input file of BIANCA. The *input file* is **automatically**

written from BIANCA for each individual at the current generation. It appears clearly that, within your model, you must provide a reading process of the design variables from that file.

The structure of the *input file* which BIANCA passes to your model is shown in Fig. 2.7.

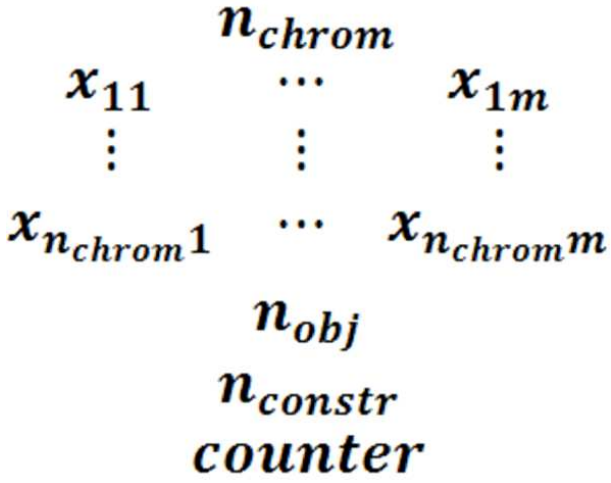

Figure 2.7: Structure of the *input file* which BIANCA passes to the external software.

As shown in Fig. 2.7, in this *input file* BIANCA automatically writes:

- the number of chromosomes, n_{chrom}, for the current individual at the current generation. In the case of an optimisation process where the number of chromosomes is fixed this value is equal to the one you declare in *line* 15 of the *.opt* input file of BIANCA, while in the case of an optimisation process where the number of chromosomes is also a variable of the problem, in this line you can find any possible integer value between the bounds that you declare in in *line* 14 of the *.opt* input file of BIANCA;

- the array of design variables for your model. The dimensions of this array are $n_{chrom} \times m$ where m is equal to the number of genes n_{gene} which you declare in *line* 16 of the *.opt* input file of BIANCA;

- the number of partial objective functions, n_{obj}, which you declare in *line* 9 of the *.opt* input file of BIANCA;

- the number of constraint functions, n_{constr}, which you declare in *line* 11 of the *.opt* input file of BIANCA;

- the *counter*. This is an integer variable which ensures the synchronisation between BIANCA and the external code. Within your model you must read the value of the *counter* from the *input file* and your model has to return this value to BIANCA by means of the creation of the *output file*, whose structure is explained in the next subsection.

As example, we show here below the structure of the *.opt* input file for BIANCA and the structure of the *input file* which BIANCA passes to the external software, in the case where the user mathematical model is realised by means of the MATLAB® package.

test_matlab.opt input file (for more details about this *.opt* input file see Sec. 3.4):

```
external
none
MATLAB
Vannucci
input_mat.txt
output_mat.txt
0
2
1
0
0
no
no 0.0
0 0
1
2
2

x_1
EXTENDED
0.0
12.556

x_2
EXTENDED
0.0
```

```
6.283

1 2
```

Structure of the *input_mat.txt* input file which BIANCA passes to MATLAB®:

```
1
1.4186705767350929    0.97096774193548396
1
0
30
```

In this simulation we have used the MATLAB® code for the construction of our mathematical model. This model is described in the example of Sec. 3.4. In this subsection we want to remark the parallelism which exists between the correct compilation of the *.opt* input file and the *input file* that BIANCA passes to the external code.

For our example, in the *test_matlab.opt* input file, we can see that the name of the MATLAB® script is *Vannucci*, as written in *line* 4; at *line* 5 we have named the input file for our MATLAB® model as *input_mat.txt*; the name of the output file which our MATLAB® model passes to BIANCA is *output_mat.txt*. Moreover, the optimisation problem described in MATLAB® environment has two design variables, i.e. *x_1* and *x_2*. In addition the structure of the genotype is made up by one chromosome with 2 genes. The problem has one objective function and no constraint function. You can see the coherence between what we have write in the *test_matlab.opt* input file for BIANCA and what BIANCA writes in the *input_mat.txt*. You can see the structure of the MATLAB® script in Sec. 3.4.

Concerning the *input_mat.txt* (input file from BIANCA to MATLAB®), for the current individual at the current generation BIANCA automatically writes the number of chromosomes, 1, the value of both design variables, $x_1 = 1.4186705767350929$ and $x_2 = 0.97096774193548396$, the number of partial objective functions, 1, the number of constraint functions, 0, and finally the value of the counter, 30.

2.6.2 The *output file* from the external code to BIANCA

As said beforehand, during the whole optimisation process, the external code passes an *output file* to BIANCA. You must include the writing operation of this file within your mathematical model (realised with your external code). The name of this file **must be the same** as the one you declare in *line* 6 of

the *.opt* input file of BIANCA. The *output file* **must be written** from your model.

The structure of the *output file* which your model passes to BIANCA is shown in Fig. 2.8.

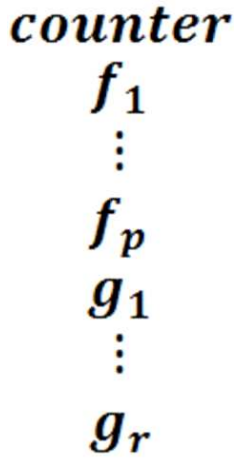

Figure 2.8: Structure of the *output file* which the external software passes to BIANCA.

As shown in Fig. 2.8, in this *output file* the following quantities must be written from your model:

- the *counter*. As previously said, this is an integer variable which ensures the synchronisation between BIANCA and the external code;
- the array of partial objective functions for your model. The dimensions of this array is n_{obj}, where n_{obj} is the number of partial objective functions which you declare in *line* 9 of the *.opt* input file of BIANCA;
- the array of constraint functions for your model. The dimensions of this array is n_{constr}, where n_{constr} is the number of constraint functions which you declare in *line* 11 of the *.opt* input file of BIANCA.

As example, we show here below the structure of the *output file* which the external software passes to BIANCA, in the case where the user mathematical model is realised by means of the MATLAB® package.

Structure of the *output_mat.txt* output file which MATLAB® passes to BIANCA:

```
30
-0.550094
```

In this simulation we have used the MATLAB® code for the construction of our mathematical model. This model is described in the example of Sec. 3.4. In this subsection we want to remark the parallelism which exists between the correct compilation of the *.opt* input file and what the *output file* (written by your model) passes to BIANCA.

For our example, the structure of the *.opt* input file is the same as the one described in the previous subsection. The name of the *.opt* input file is *test_matlab.opt*. The name of the output file which our MATLAB® model passes to BIANCA is *output_mat.txt*, as written in *line* 6.

You can see that in the *output_mat.txt* we can find the value of the counter, 30, and the value of the objective function, -0.550094. Since our optimisation problem is an unconstrained problem we cannot write the value of constraint functions in the *output_mat.txt* file.

Chapter 3

Examples

3.1 Test function example: welded beam design problem

In this section we show the input and output files concerning a particular test case optimisation problem: the welded beam design problem. This problem was firstly studied by Rao. The objective is to design a welded beam for minimum cost subject to several constraints, e.g. on shear stress, bending stress, buckling load, deflection of the beam and other side constraints. There are 4 design variables: the height of the weld $h(x_1)$, the length of the weld $l(x_2)$, and finally the height $t(x_3)$ and the width $b(x_4)$ of the beam.

Mathematically, the problem can be stated as follows:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) = 1.10471{x_1}^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2) \\ &\text{subject to :} \begin{cases} g_1(\mathbf{x}) = \tau(\mathbf{x}) - 13000 \leq 0 \\ g_2(\mathbf{x}) = \sigma(\mathbf{x}) - 30000 \leq 0 \\ g_3(\mathbf{x}) = x_1 - x_4 \leq 0 \\ g_4(\mathbf{x}) = 0.10471{x_1}^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \leq 0 \\ g_5(\mathbf{x}) = 0.125 - x_1 \leq 0 \\ g_6(\mathbf{x}) = \delta(\mathbf{x}) - 0.25 \leq 0 \\ g_7(\mathbf{x}) = 6000 - P_c(\mathbf{x}) \leq 0 \end{cases} \end{aligned} \tag{3.1}$$

where:

$$\begin{cases} \tau(\mathbf{x}) = \sqrt{(\tau_1)^2 + 2\tau_1\tau_2\frac{x_2}{2R} + (\tau_2)^2} \\ \tau_1 = \dfrac{6000}{\sqrt{2}x_1x_2} \\ \tau_2 = \dfrac{MR}{J} \\ M = 6000\left(14 + \dfrac{x_2}{2}\right) \\ R = \sqrt{\dfrac{{x_2}^2}{4} + \left(\dfrac{x_1 + x_3}{2}\right)^2} \\ J = 2\left\{\sqrt{2}x_1x_2\left[\dfrac{{x_2}^2}{12} + \left(\dfrac{x_1 + x_3}{2}\right)^2\right]\right\} \\ \sigma(\mathbf{x}) = \dfrac{504000}{{x_3}^2x_4} \\ \delta(\mathbf{x}) = \dfrac{2.1952}{{x_3}^2x_4} \\ P_c(\mathbf{x}) = 64746.022\left(1 - 0.0282346x_3\right)x_3{x_4}^3 \end{cases} \tag{3.2}$$

The name of the job session is *welded_beam*, so the input files must have the same name with extensions *.gen* and *.opt* and the 3 output files are going to have the same name with extensions *.bio*, *.pop* and *.sta*. For this case we perform an optimisation process with 2 population evolving at the same time.

The *genetic parameters* are:

- $n_{pop} = 2$;
- $n_{ind} = 400$;
- *stop crit.* = *fixed_generations*;
- $n_{gen} = 200$;
- $p_{cross} = 0.85$;
- $p_{mut} = 0.0025$;
- $p_{shift} = 0.0$;
- $p_{mutchrom} = 0.0$;
- *selection operator* = roulette wheel;

- *selection pressure* = 1.0;
- *elitism operator* = active;
- *isolation time* = 50.

We show here below the structure of the *welded_beam.gen* input file:

```
2
400
fixed_generations
200
0.85
0.0025
0.0
0.0
1
1.0
1
50
```

The *optimisation parameters* are:

- *simulation environment type* = internal;
- *kind of problem* = test_function;
- *external code* = none;
- *external code model file* = none;
- *external code input file* = none;
- *external code output file* = none;
- *function ID* = 80;
- *minimisation or maximisation* = minimisation;
- $n_{obj} = 1$;
- *constr. active or not* = active;
- $n_{constr} = 7$;
- $CHROMVAR$ = no;

- $CHROMMIN\ exp =$ no 0.0;
- $n_{chrommin}\ n_{chrommax} = 0\ 0$;
- $n_{chrom} = 1$;
- $n_{gene} = 4$;
- $n_{var} = 4$;
- *variables*:
  - *x_1*, extended, bounds [0.1 2.0]mm;
  - *x_2*, extended, bounds [0.1 10.0]mm;
  - *x_3*, extended, bounds [0.1 10.0]mm;
  - *x_4*, extended, bounds [0.1 2.0]mm;

Then we show here below the structure of the *welded_beam.opt* input file:

```
internal
test_function
none
none
none
none
80
2
1
1
7
no
no 0.0
0 0
1
4
4

x_1
EXTENDED
0.1
2.0
```

x_2
EXTENDED
0.1
10.0

x_3
EXTENDED
0.1
10.0

x_4
EXTENDED
0.1
2.0

1 2 3 4

You can see the structure of the *welded_beam.bio*, *welded_beam.pop* and *welded_beam.sta* output files in the following paths:

BIANCA 3.1 User guide\ Examples\ welded_beam.bio;
BIANCA 3.1 User guide\ Examples\ welded_beam.pop;
BIANCA 3.1 User guide\ Examples\ welded_beam.sta.

We perform also the post processing of results and we show here below the structure of the *post_processing.inp* input file:

MATLAB
linear
off
png

In this case we obtain two image files saved as Portable Network Graphics files named *obj_min_welded_beam.png* and *obj_mean_welded_beam.png*. You can find those files in the following paths:

BIANCA 3.1 User guide\ Examples\ obj_min_welded_beam.png;
BIANCA 3.1 User guide\ Examples\ obj_mean_welded_beam.png.

Figs. 3.1 and 3.2 show the curves of the best feasible solution vs generations and the average value of the objective function vs generations, respectively.

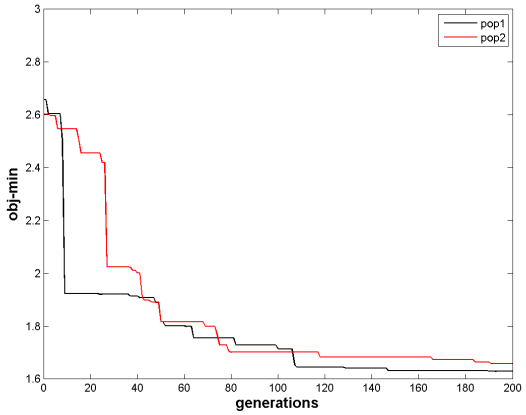

Figure 3.1: Best feasible solution vs generations for the welded beam design problem.

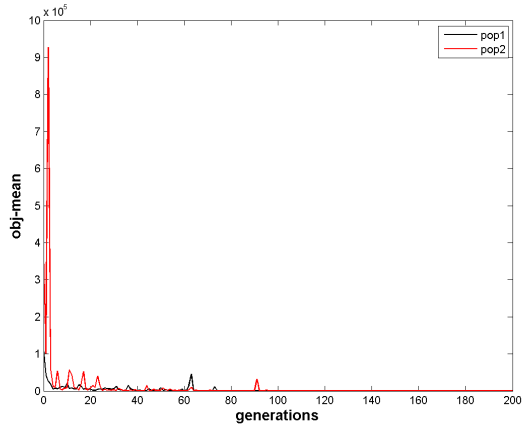

Figure 3.2: Average value of the objective function vs generations for the for the welded beam design problem.

3.2 Library function example: laminates' design problem

In this section we show the input and output files concerning the optimisation process on the laminate's elastic properties design problem. For more details about this kind of problem see [1, 7, 3]. Here we want to find a laminate that has the following symmetries:

- elastic uncoupling;
- in-plane stiffness isotropy;
- bending stiffness orthotropy.

We make 2 kind of simulations: in the first one the number of layers is fixed and only the orientations are the design variables, while in the second one the number of plies is variable and the design variables are the orientation and the thickness of each layer. In this last case the reproduction between different species is required. In the following subsections we show how to perform the optimisation process by means of the library functions for the design of laminate elastic symmetries developed within BIANCA.

3.2.1 Fixed number of chromosomes/plies

In this first simulation we use the library function with ID 71.
The name of the job session is *Symmetries*, so the input files must have the same name with extensions *.gen* and *.opt* and the 3 output files are going to have the same name with extensions *.bio*, *.pop* and *.sta*. In addition we must compile in a correct way the *library.inp* input file.

The *genetic parameters* are:

- $n_{pop} = 1$;
- $n_{ind} = 500$;
- *stop crit.* = *fixed_generations*;
- $n_{gen} = 500$;
- $p_{cross} = 0.85$;
- $p_{mut} = 0.002$;
- $p_{shift} = 0.0$;

- $p_{mutchrom} = 0.0$;
- *selection operator* = roulette wheel;
- *selection pressure* = 1.0;
- *elitism operator* = active;
- *isolation time* = 0.

We show here below the structure of the *Symmetries.gen* input file:

```
1
500
fixed_generations
500
0.85
0.002
0.0
0.0
1
1.0
1
0
```

For our example, the laminate has 14 plies and the only variables are the layers' orientations. According to what we have already explained in Sec. 2.3 about the *library.inp* input file and about the structure of individual's genotype, in this case the number of variables is 13 (the first ply has a fixed orientation, i.e. 0.0°). Each ply corresponds to a chromosome in the structure of the genotype and the layer's orientation is linked to a single gene. Then, the *optimisation parameters* are:

- *simulation environment type* = internal;
- *kind of problem* =library function;
- *external code* = none;
- *external code model file* = none;
- *external code input file* = none;
- *external code output file* = none;

- *function ID* = 71;
- *minimisation or maximisation* = minimisation;
- $n_{obj} = 3$;
- *constr. active or not* = not active;
- $n_{constr} = 0$;
- $CHROMVAR$ = no;
- $CHROMMIN$ *exp* = no 0.0;
- $n_{chrommin}$ $n_{chrommax}$ = 0 0;
- $n_{chrom} = 13$;
- $n_{gene} = 1$;
- $n_{var} = 1$;
- *variables*:
  - *angle*, regular discrete, step 1.0°, bounds [−90.0° 90.0°];

We show here below the structure of the *Symmetries.opt* input file:

```
internal
library_function
none
none
none
none
71
2
3
0
0
no
no 0.0
0 0
13
1
1
```

```
angle
REGULAR_DISCR
-90.0
90.0
1.0

1
```

The structure of the *library.inp* input file is the following:

```
membrane_stiffness_isotropy
bending_stiffness_orthotropy_K=0
uncoupling
```

You can see the structure of the *Symmetries.bio*, *Symmetries.pop* and *Symmetries.sta* output files in the following paths:

BIANCA 3.1 User guide\ Examples\ Symmetries.bio
BIANCA 3.1 User guide\ Examples\ Symmetries.pop
BIANCA 3.1 User guide\ Examples\ Symmetries.sta.

We perform also the post processing of results and we show here below the structure of the *post_processing.inp* input file:

```
MATLAB
semilog
off
png
```

In this case we obtain two image files saved as Portable Network Graphics files named *obj_min_Symmetries.png* and *obj_mean_Symmetries.png*. You can find those files in the following paths:

BIANCA 3.1 User guide\ Examples\ obj_min_Symmetries.png;
BIANCA 3.1 User guide\ Examples\ obj_mean_Symmetries.png.

Figs. 3.3 and 3.4 show the curves of the best feasible solution vs generations and the average value of the objective function vs generations, respectively.

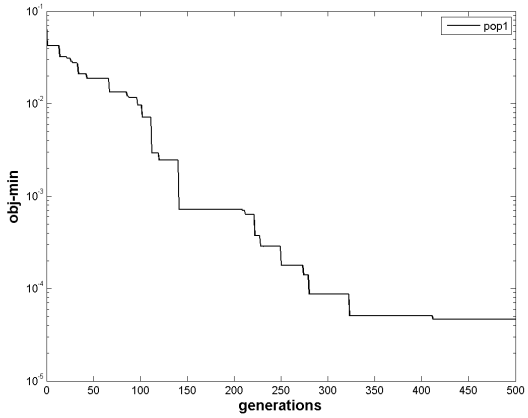

Figure 3.3: Best feasible solution vs generations for the design of laminate's elastic symmetries.

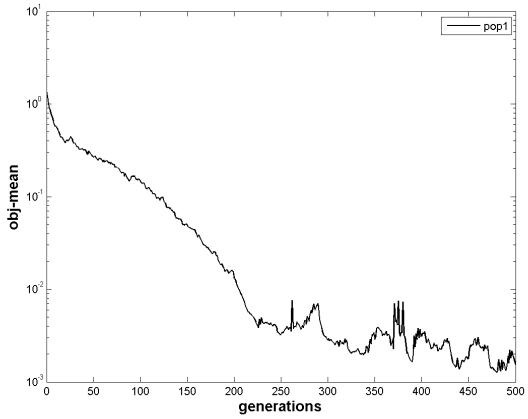

Figure 3.4: Average value of the objective function vs generations for the design of laminate's elastic symmetries.

3.2.2 Variable number of chromosomes/plies

In this second simulation we use the library function with ID 72.
The name of the job session is *Symmetries_var*, so the input files must have the same name with extensions *.gen* and *.opt* and the 3 output files are going to have the same name with extensions *.bio*, *.pop* and *.sta*. In addition we must compile in a correct way the *library.inp* input file.

The *genetic parameters* are:

- $n_{pop} = 1$;
- $n_{ind} = 500$;
- *stop crit.* = *fixed_generations*;
- $n_{gen} = 500$;
- $p_{cross} = 0.85$;
- $p_{mut} = 0.002$;
- $p_{shift} = 0.5$;
- $p_{mutchrom} = 0.008$;
- *selection operator* = roulette wheel;
- *selection pressure* = 1.0;
- *elitism operator* = active;
- *isolation time* = 0.

We show here below the structure of the *Symmetries_var.gen* input file:

```
1
500
fixed_generations
500
0.85
0.002
0.5
0.008
1
1.0
```

1
0

For our example, the laminate has a variable number of plies and the design variables are the orientation and thickness of every layer. According to what we have already explained in Sec. 2.3 about the *library.inp* input file and about the structure of individual's genotype, in this case the number of variables is $2n$ where n is the number of chromosomes. We consider that the number of chromosome can varies between 12 and 16. Each ply corresponds to a chromosome in the structure of the genotype and the orientation and thickness of each layer are linked to two different genes. Then, the *optimisation parameters* are:

- *simulation environment type* = internal;
- *kind of problem* =library function;
- *external code* = none;
- *external code model file* = none;
- *external code input file* = none;
- *external code output file* = none;
- *function ID* = 72;
- *minimisation or maximisation* = minimisation;
- $n_{obj} = 3$;
- *constr. active or not* = not active;
- $n_{constr} = 0$;
- $CHROMVAR$ = yes;
- $CHROMMIN$ *exp* = yes 2.0;
- $n_{chrommin}$ $n_{chrommax}$ = 12 16;
- $n_{chrom} = 0$;
- $n_{gene} = 2$;
- $n_{var} = 2$;

- *variables*:
  - *angle*, regular discrete, step 1.0°, bounds [−90.0° 90.0°];
  - *thick*, extended, bounds [0.1 0.2]mm.

We show here below the structure of the *Symmetries_var.opt* input file:

```
internal
library_function
none
none
none
none
72
2
3
0
0
yes
yes 2.0
12 16
0
1
1

angle
REGULAR_DISCR
-90.0
90.0
1.0

thick
EXTENDED
0.1
0.2

1 2
```

The structure of the *library.inp* input file is the following:

```
membrane_stiffness_isotropy
```

bending_stiffness_orthotropy_K=0
uncoupling

You can see the structure of the *Symmetries_var.bio*, *Symmetries_var.pop* and *Symmetries_var.sta* output files in the following paths:

BIANCA 3.1 User guide\ Examples\ Symmetries_var.bio
BIANCA 3.1 User guide\ Examples\ Symmetries_var.pop
BIANCA 3.1 User guide\ Examples\ Symmetries_var.sta.

We perform also the post processing of results and we show here below the structure of the *post_processing.inp* input file:

MATLAB
semilog
off
png

In this case we obtain two image files saved as Portable Network Graphics files named *obj_min_Symmetries_var.png* and *obj_mean_Symmetries_var.png*. You can find those files in the following paths:

BIANCA 3.1 User guide\ Examples\ obj_min_Symmetries_var.png;
BIANCA 3.1 User guide\ Examples\ obj_mean_Symmetries_var.png.

Figs. 3.5 and 3.6 show the curves of the best feasible solution vs generations and the average value of the objective function vs generations, respectively.

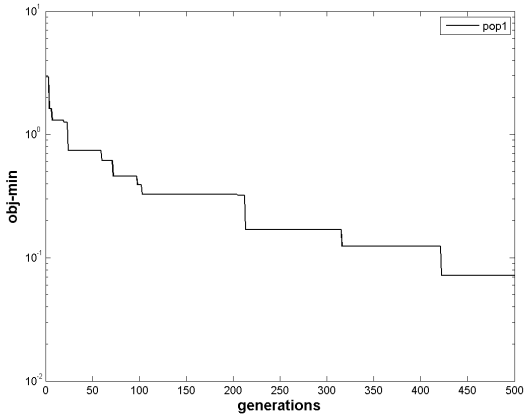

Figure 3.5: Best feasible solution vs generations for the design of laminate's elastic symmetries, variable number of layers.

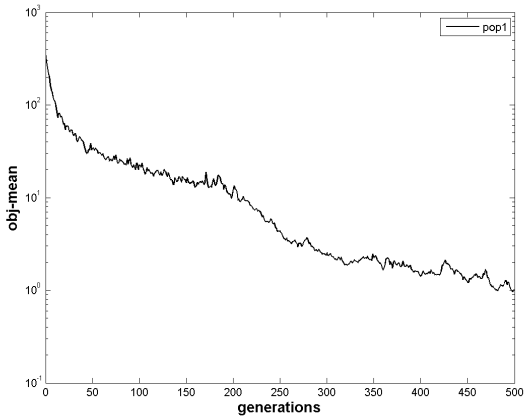

Figure 3.6: Average value of the objective function vs generations for the design of laminate's elastic symmetries, variable number of layers.

3.3 User-defined model example: beam design problem

In this section we show the input and output files concerning the optimisation process on a possible user-defined model and also how to write the mathematical model in the MACRO_MY_PROBLEM.f95.

In this example we consider a simple optimisation problem about beams' design. We have a clamped beam subject to a tip load P of 1000 N. The beam is made up by an Aluminium alloy with a Young modulus E equal to 72 GPa and a tensile yield stress σ_y of 345 MPa. The beam has a circular section. The design variables are the beam's length L and the beam's radius R.
We want to find the optimal values of L and R which minimise the tip displacement δ and respect both constraints imposed on the maximum stress and on the beam's volume. The reference value of the volume V_{ref} is 140000 mm^3.

The problem can be formulated as follows:

$$\begin{aligned} &\min_{L,R} \delta(L,R) = \frac{4PL^3}{3\pi E R^4} \\ &\text{subject to}: \begin{cases} \dfrac{4PL}{\pi R^3} - \sigma_y \leq 0 \\ L\pi R^2 - V_{ref} \leq 0 \end{cases} \end{aligned} \tag{3.3}$$

The name of the job session is *beam design*, so the input files must have the same name with extensions *.gen* and *.opt* and the 3 output files are going to have the same name with extensions *.bio*, *.pop* and *.sta*.
Since the problem do not require the cross-over between species, the structure of the individual's genotype is made up by a single chromosome with two genes which identify the design variables, i.e. the length L and the radius R. In addition, since the number of chromosome is fixed we must write our model by means of the subroutine *my_problem*. We have written our model in the MACRO_MY_PROBLEM.f95 as shown in Fig. 3.7.

```
!     ****************************************************************
!                          MACRO MY PROBLEM
!     ****************************************************************
      subroutine my_problem(npop,nind,nchrom,ngene,x_being,           &
     &                      n_obj,obj,constr_active,                  &
     &                      n_constr,constr_ineq)
      implicit none
      integer npop,nind,nchrom,ngene
      integer n_obj,n_constr,constr_active
      double precision,dimension(10,2000,50,50) :: x_being
      double precision,dimension(10,2000,30) :: obj
      double precision,dimension(10,2000,30) :: constr_ineq
!     VARIABLE DECLARATION FOR OUR PROBLEM
!     beam design problem
      double precision L,R,delta,sigma_max,V
      integer i,j,k
      double precision,parameter ::                                   &
     & E=0.72000D+06,pi=4.0d0*datan(1.0d0),P=1000.0d0,                &
     & sigma_n=345.0d0,V_ref= 140000.0d0

!     L = x_1 1st DESIGN VARIABLE
!     R = x_2 2nd DESIGN VARIABLE
!     delta = objective function
!     sigma_max-sigma_n = constraint n. 1
!     V-V_ref = constraint n. 2
       do i=1,npop
          do j=1,nind
                L = x_being(i,j,1,1)
                R = x_being(i,j,1,2)
!     writing the expression of delta
               delta=4.0*P*(L**3.0)/(3.0*pi*E*R**(4.0))
!     writing the expression of sigma_max
               sigma_max=4.0*P*L/(pi*R**(3.0))
!     writing the expression of V
               V=pi*L*(R**2.0)
!     assigning the expression of delta to the objective function
                  obj(i,j,1)=delta
!     assigning the expression of sigma_max-sigma_n to the constraint
                  constr_ineq(i,j,1)=sigma_max-sigma_n
!     assigning the expression of V-V_ref to the constraint
                  constr_ineq(i,j,2)=V-V_ref
          end do
       end do
      end subroutine my_problem

```

Figure 3.7: Structure of the *macro MACRO_MY_PROBLEM.f95* for the beam design problem.

The *genetic parameters* are:

- $n_{pop} = 1$;
- $n_{ind} = 100$;
- *stop crit.* = *fixed_generations*;
- $n_{gen} = 200$;
- $p_{cross} = 0.85$;
- $p_{mut} = 0.01$;
- $p_{shift} = 0.0$;
- $p_{mutchrom} = 0.0$;
- *selection operator* = roulette wheel;
- *selection pressure* = 1.0;
- *elitism operator* = active;
- *isolation time* = 0.

Then we show here below the structure of the *beam design.gen* input file:

```
1
100
fixed_generations
200
0.85
0.01
0.0
0.0
1
1.0
1
0
```

The *optimisation parameters* are:

- *simulation environment type* = internal;
- *kind of problem* = my problem;

- *external code* = none;
- *external code model file* = none;
- *external code input file* = none;
- *external code output file* = none;
- *function ID* = 0;
- *minimisation or maximisation* = minimisation;
- $n_{obj} = 1$;
- *constr. active or not* = active;
- $n_{constr} = 2$;
- $CHROMVAR$ = no;
- $CHROMMIN$ *exp* = no 0.0;
- $n_{chrommin}$ $n_{chrommax}$ = 0 0;
- $n_{chrom} = 1$;
- $n_{gene} = 2$;
- $n_{var} = 2$;
- *variables*:
  - *Length*, regular discrete, step 1.0 mm, bounds [100.0 500.0]mm;
  - *Radius*, regular discrete, step 1.0 mm, bounds [10.0 50.0]mm.

Then we show here below the structure of the *beam design.opt* input file:

```
internal
my_problem
none
none
none
none
0
2
1
```

```
1
2
no
no 0.0
0 0
1
2
2

Length
REGULAR_DISCR
100.0
500.0
1.0

Radius
REGULAR_DISCR
10.0
50.0
1.0

1 2
```

You can see the structure of the *beam design.bio*, *beam design.pop* and *beam design.sta* output files in the following paths:

```
BIANCA 3.1 User guide\ Examples\ beam design.bio;
BIANCA 3.1 User guide\ Examples\ beam design.pop;
BIANCA 3.1 User guide\ Examples\ beam design.sta.
```

We perform also the post processing of results and we show here below the structure of the *post_processing.inp* input file:

```
MATLAB
linear
off
png
```

In this case we obtain two image files saved as Portable Network Graphics files named *obj_min_beam design.png* and *obj_mean_beam design.png*. You can find those files in the following paths:

BIANCA 3.1 User guide\ Examples\ obj_min_beam design.png;
BIANCA 3.1 User guide\ Examples\ obj_mean_beam design.png.

Figs. 3.8 and 3.9 show the curves of the best feasible solution vs generations and the average value of the objective function vs generations, respectively.

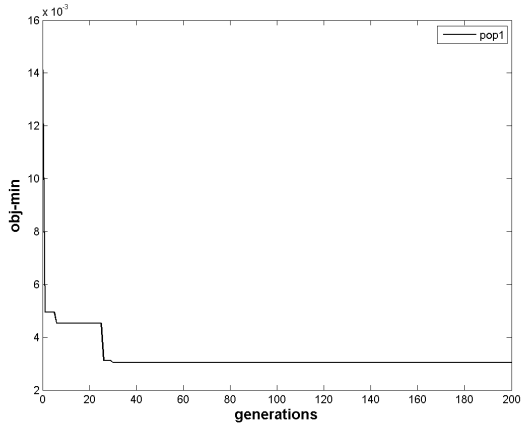

Figure 3.8: Best feasible solution vs generations for the beam design problem.

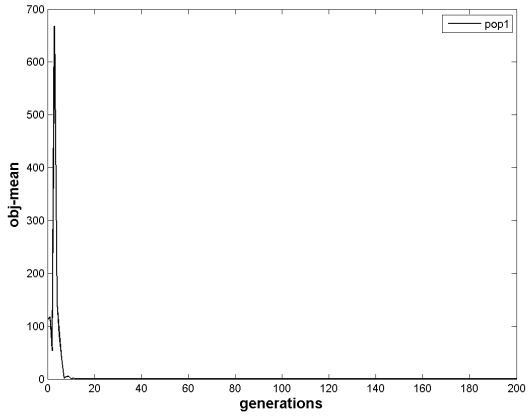

Figure 3.9: Average value of the objective function vs generations for the beam design problem.

3.4 Example of interface with MATLAB® code: Vannucci's function problem

In order to have a general idea on how BIANCA 3.1 works when it is interfaced with external codes, we show here the input and output files concerning the optimisation process on the unconstrained Vannucci's function problem. The problem can be stated as follows:

$$\begin{cases} \min f(x_1, x_2) = -e^{ka\sqrt{x_1^2+x_2^2}} \sin(ax_1)\cos(2bx_2) \\ 0 \le x_1 \le 4\pi \\ 0 \le x_2 \le 2\pi \end{cases} \tag{3.4}$$

Although this problem is already implemented as test function within BIANCA, we have rewrite this problem in MATLAB® environment in order to explain how you can easily interface with BIANCA any model realised within the MATLAB® code.

First of all we can consider the structure of the model written within MATLAB®. The structure is completely free, except for the part concerning the exchange of data with BIANCA. For our example the model is written as MATLAB® script whose name is *Vannucci.m* and whose structure is shown in Fig 3.10. The data exchange between BIANCA and our model is realised according to what we have explained in Sec. 2.6.

You can see the structure of this MATLAB® script in the following path:

BIANCA 3.1 User guide\ Examples\ Vannucci.m;

The name of the job session is *test_matlab*, so the input files must have the same name with extensions *.gen* and *.opt* and the 3 output files are going to have the same name with extensions *.bio*, *.pop* and *.sta*.

The *genetic parameters* are:

- $n_{pop} = 1$;
- $n_{ind} = 100$;
- *stop crit.* = *fixed_generations*;
- $n_{gen} = 200$;
- $p_{cross} = 0.85$;
- $p_{mut} = 0.01$;

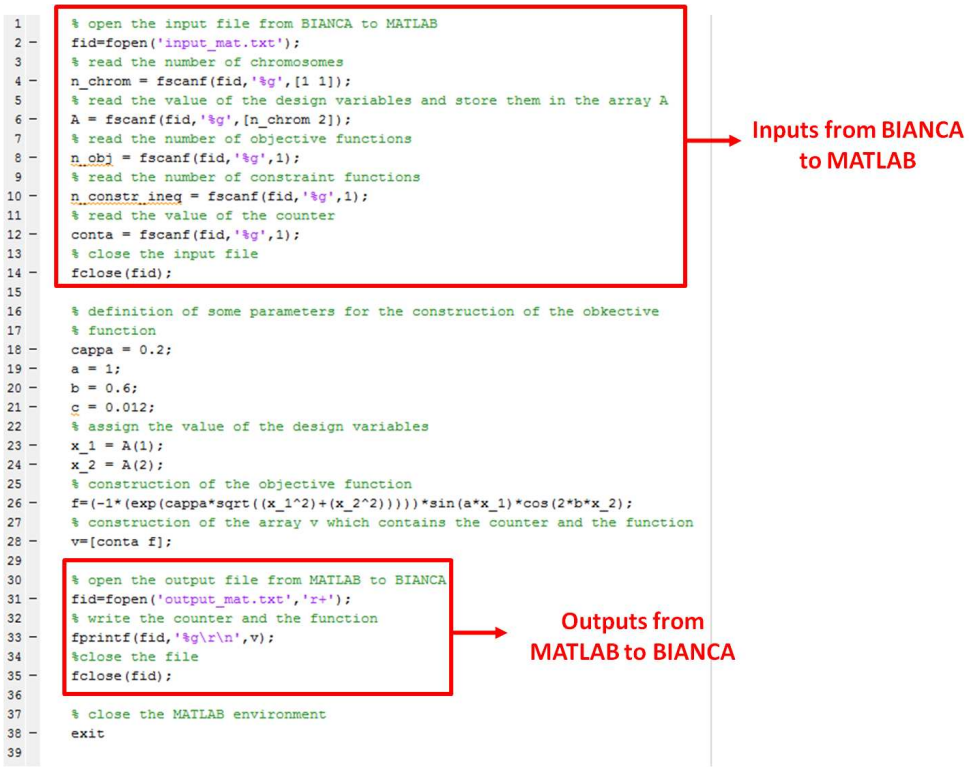

Figure 3.10: Structure of the Vannucci.m MATLAB® script.

- $p_{shift} = 0.0$;
- $p_{mutchrom} = 0.0$;
- *selection operator* = roulette wheel;
- *selection pressure* = 1.0;
- *elitism operator* = active;
- *isolation time* = 0.

Then we show here below the structure of the *test_matlab.gen* input file:

1
100
fixed_generations
200
0.85
0.01

0.0
0.0
1
1.0
1
0

The *optimisation parameters* are:

- *simulation environment type* = external;
- *kind of problem* = none;
- *external code* = MATLAB;
- *external code model file* = Vannucci;
- *external code input file* = input_mat.txt;
- *external code output file* = output_mat.txt;
- *function ID* = 0;
- *minimisation or maximisation* = minimisation;
- $n_{obj} = 1$;
- *constr. active or not* = not active;
- $n_{constr} = 0$;
- $CHROMVAR$ = no;
- $CHROMMIN$ *exp* = no 0.0;
- $n_{chrommin}$ $n_{chrommax} = 0$ 0;
- $n_{chrom} = 1$;
- $n_{gene} = 2$;
- $n_{var} = 2$;
- *variables*:
  - *x_1*, extended, bounds [0.0 12.556];
  - *x_2*, extended, bounds [0.0 6.283];

Then we show here below the structure of the *test_matlab.opt* input file:

```
external
none
MATLAB
Vannucci
input_mat.txt
output_mat.txt
0
2
1
0
0
no
no 0.0
0 0
1
2
2

x_1
EXTENDED
0.0
12.556

x_2
EXTENDED
0.0
6.283

1 2
```

You can see the structure of the *test_matlab.bio*, *test_matlab.pop* and *test_matlab.sta* output files in the following paths:

BIANCA 3.1 User guide\ Examples\ test_matlab.bio;
BIANCA 3.1 User guide\ Examples\ test_matlab.pop;
BIANCA 3.1 User guide\ Examples\ test_matlab.sta.

We perform also the post processing of results and we show here below the structure of the *post_processing.inp* input file:

MATLAB
linear
off
png

In this case we obtain two image files saved as Portable Network Graphics files named *obj_min_test_matlab.png* and *obj_mean_test_matlab.png*. You can find those files in the following paths:

BIANCA 3.1 User guide\ Examples\ obj_min_test_matlab.png;
BIANCA 3.1 User guide\ Examples\ obj_mean_test_matlab.png.

Figs. 3.11 and 3.12 show the curves of the best feasible solution vs generations and the average value of the objective function vs generations, respectively.

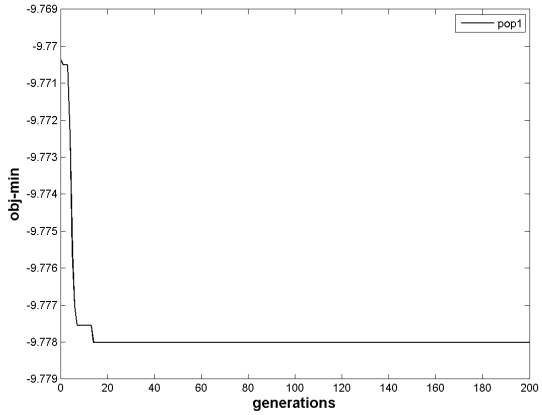

Figure 3.11: Best feasible solution vs generations for the unconstrained Vannucci's function problem.

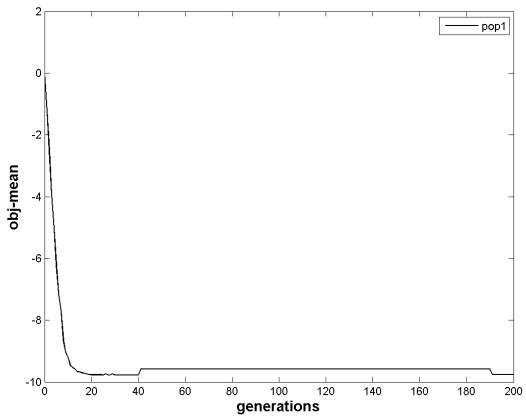

Figure 3.12: Average value of the objective function vs generations for the unconstrained Vannucci's function problem.

3.5 Example of interface with ANSYS® code: stiffened plate design problem

In order to have a general idea on how BIANCA 3.1 works when it is interfaced with external codes, we show here the input and output files concerning the optimisation process on the engineering problem of the optimal design of an aeronautical stiffened panel. Here, the optimal design of the stiffened panel is intended in terms of minimizing its weight and, at the same time, respecting a constraint imposed on its buckling load. The problem to obtain a minimum weight panel which presents particular characteristics in terms of buckling load, is a very important issue in the aeronautics.

We show here that this optimisation problem can be formulated and solved in a very general way by applying our approach for the optimisation of modular engineering systems where not only the characteristics of each module, but also the number of the constitutive modules are taken into account as optimisation variables. In addition, this problem is numerically more complex than the ones presented in the previous sections: in fact it is a constrained optimisation problem where either the objective or the constraint function are evaluated by means of a Finite Element (FE) code. To represent the wing-box section we use the ANSYS® FE code. The optimisation variables are: the number of stiffeners n, and for each stiffener there are two design variables, i.e. the thickness t_{si} and the height h_i, $(i = 1...n)$. The total number of design variables is $2n$.

Concerning the mathematical formalisation of the constrained optimisation problem, it can be stated as:

$$\begin{cases} \min\limits_{n,\mathbf{t_s},\mathbf{h}} \left[W^{up}_{stiff}\left(n,\mathbf{t_s},\mathbf{h}\right)\right] \\ \text{subject to}: \quad p_{ref} - p_{cr}\left(n,\mathbf{t_s},\mathbf{h}\right) \leq 0 \end{cases} \tag{3.5}$$

In Eq.(3.5), $W^{up}_{stiff}\left(n,\mathbf{t_s},\mathbf{h}\right)$ stands for the weight of the upper panel's stiffeners, while the quantities $n, \mathbf{t_s}, \mathbf{h}$ represent the number of stiffeners for each panel and the vectors of stiffeners' thickness and heights respectively, i.e. the design variables. The quantity $p_{cr}\left(n,\mathbf{t_s},\mathbf{h}\right)$ stands for the *first buckling load* of the structure, while p_{ref} is a reference value for this quantity. More details about this problem can be found in [4].

First of all we can consider the structure of the model written within ANSYS®. The structure is completely free, except for the part concerning the exchange of data with BIANCA. As well explained in Sec. 2.6 the interface between the two codes is realised by means of the creation of two input/output files which pass the design variables from BIANCA to ANSYS®

and the objective and constraints from ANSYS® to BIANCA. For our case the name of the ANSYS® file which contains all the information about the wing-box FE model is *panel.lgw*. You can see the structure of this file in the following path:

BIANCA 3.1 User guide\ Examples\ panel.lgw;

The structure of this file is substantially articulated in three parts: the first one concerns the reading operations of design variables that BIANCA passes to the FE model, the second one concerns the creation of the model and, finally, the third one concerns the writing operations of the output file that contains the counter, objective function and constraint that ANSYS® passes to BIANCA.

The name of the job session is *test_ansys*, so the input files must have the same name with extensions *.gen* and *.opt* and the 3 output files are going to have the same name with extensions *.bio*, *.pop* and *.sta*.

The *genetic parameters* are:

- $n_{pop} = 1$;
- $n_{ind} = 150$;
- *stop crit.* = *fixed_generations*;
- $n_{gen} = 200$;
- $p_{cross} = 0.85$;
- $p_{mut} = 0.0067$;
- $p_{shift} = 0.5$;
- $p_{mutchrom} = 0.04$;
- *selection operator* = tournament;
- *selection pressure* = 1.0;
- *elitism operator* = active;
- *isolation time* = 0.

Then we show here below the structure of the *test_ansys.gen* input file:

1

150
fixed_generations
200
0.85
0.0067
0.5
0.04
2
1.0
1
0

The *optimisation parameters* are:

- *simulation environment type* = external;
- *kind of problem* = none;
- *external code* = ANSYS;
- *external code model file* = panel.lgw;
- *external code input file* = inputs.txt;
- *external code output file* = outputs.txt;
- *function ID* = 0;
- *minimisation or maximisation* = minimisation;
- $n_{obj} = 1$;
- *constr. active or not* = active;
- $n_{constr} = 1$;
- $CHROMVAR$ = yes;
- $CHROMMIN$ *exp* = no 0.0;
- $n_{chrom_{min}}$ $n_{chrom_{max}}$ = 20 26;
- $n_{chrom} = 0$;
- $n_{gene} = 2$;

- $n_{var} = 2$;
- *variables*:
  - *thick*, regular discrete, step 0.1 mm, bounds [2.0 5.0]mm;
  - *height*, regular discrete, step 0.5 mm, bounds [50.0 100.0]mm;

Then we show here below the structure of the *test_ansys.opt* input file:

```
external
none
ANSYS
panel.lgw
inputs.txt
outputs.txt
0
2
1
1
1
yes
no 0.0
20 26
0
2
2

thick
REGULAR_DISCR
2.0
5.0
0.1

height
REGULAR_DISCR
50.0
100.0
0.5

1 2
```

You can see the structure of the *test_ansys.bio*, *test_ansys.pop* and *test_ansys.sta* output files in the following paths:

BIANCA 3.1 User guide\ Examples\ test_ansys.bio;
BIANCA 3.1 User guide\ Examples\ test_ansys.pop;
BIANCA 3.1 User guide\ Examples\ test_ansys.sta.

We perform also the post processing of results and we show here below the structure of the *post_processing.inp* input file:

MATLAB
linear
off
png

In this case we obtain two image files saved as Portable Network Graphics files named *obj_min_test_ansys.png* and *obj_mean_test_ansys.png*. You can find those files in the following paths:

BIANCA 3.1 User guide\ Examples\ obj_min_test_ansys.png;
BIANCA 3.1 User guide\ Examples\ obj_mean_test_ansys.png.

Figs. 3.13 and 3.14 show the curves of the best feasible solution vs generations and the average value of the objective function vs generations, respectively.

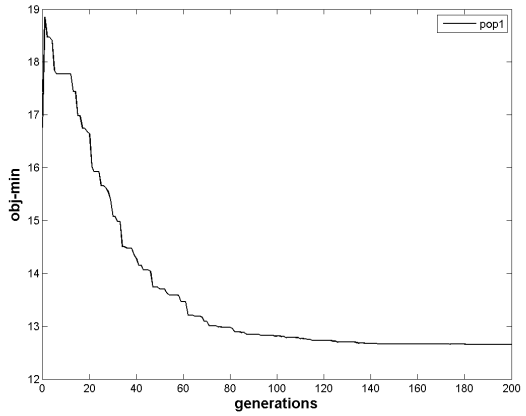

Figure 3.13: Best feasible solution vs generations for the stiffened plate design problem.

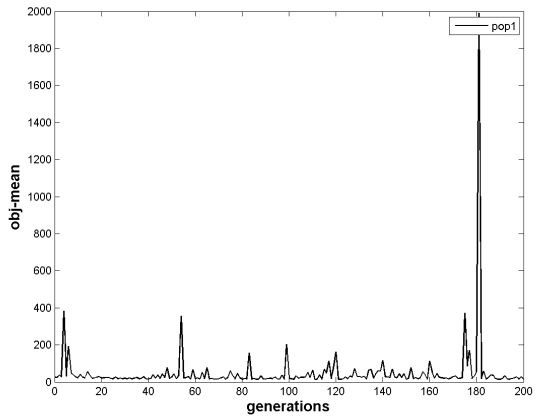

Figure 3.14: Average value of the objective function vs generations for the stiffened plate design problem.

3.6 Example of interface with ABAQUS® code: beam design problem

In this section we consider a simple optimisation problem about beams' design. We have a clamped beam subject to a tip load P of 1000 N. The beam is made up by a Steel alloy with a Young modulus E equal to 200 GPa. The beam has a rectangular cross section. The design variables are the beam's length L and the beam's height h. The width b of the section is equal to 10 mm. The model of the beam is realised by means of the ABAQUS® FE code.
We want to find the optimal values of L and h which minimise the absolute value of the tip displacement $|\delta|$.

The problem can be formulated as follows:

$$\min_{L,h} |\delta(L,h)| \tag{3.6}$$

First of all we must consider the structure of the model written within ABAQUS®. Again, the structure is completely free, except for the part concerning the exchange of data with BIANCA. As well explained in Sec. 2.6 the interface between the two codes is realised by means of the creation of two input/output files which pass the design variables from BIANCA to ABAQUS® and the objective and constraints from ABAQUS® to BIANCA. For our case the name of the ABAQUS® file which contains all the information about the beam FE model is *cantilevered_beam.py*. You can see the structure of this file in the following path:

BIANCA 3.1 User guide\ Examples\ cantilevered_beam.py;

The structure of this file is substantially articulated in three parts: the first one concerns the reading operations of design variables that BIANCA passes to the FE model, the second one concerns the creation of the model and, finally, the third one concerns the writing operations of the output file that contains the counter, objective function and constraint that ABAQUS® passes to BIANCA.

The name of the job session is *test_abaqus*, so the input files must have the same name with extensions *.gen* and *.opt* and the 3 output files are going to have the same name with extensions *.bio*, *.pop* and *.sta*.

The *genetic parameters* are:

- $n_{pop} = 1$;

- $n_{ind} = 30$;
- *stop crit.* = *fixed_generations*;
- $n_{gen} = 100$;
- $p_{cross} = 0.85$;
- $p_{mut} = 0.033$;
- $p_{shift} = 0.0$;
- $p_{mutchrom} = 0.0$;
- *selection operator* = tournament;
- *selection pressure* = 1.0;
- *elitism operator* = active;
- *isolation time* = 0.

Then we show here below the structure of the *test_abaqus.gen* input file:

```
1
30
fixed_generations
100
0.85
0.033
0.0
0.0
2
1.0
1
0
```

The *optimisation parameters* are:

- *simulation environment type* = external;
- *kind of problem* = none;
- *external code* = ABAQUS;
- *external code model file* = cantilevered_beam.py;

- *external code input file* = input.txt;
- *external code output file* = out.txt;
- *function ID* = 0;
- *minimisation or maximisation* = minimisation;
- $n_{obj} = 1$;
- *constr. active or not* = not active;
- $n_{constr} = 0$;
- $CHROMVAR$ = no;
- $CHROMMIN$ *exp* = no 0.0;
- $n_{chrommin}$ $n_{chrommax}$ = 0 0;
- $n_{chrom} = 1$;
- $n_{gene} = 2$;
- $n_{var} = 2$;
- *variables*:
  - *Length*, extended, bounds [500.0 1000.0]mm;
  - *Height*, extended, bounds [10.0 50.0]mm;

Then we show here below the structure of the *test_abaqus.opt* input file:

```
external
none
ABAQUS
cantilevered_beam.py
input.txt
out.txt
0
2
1
0
0
no
```

```
no 0.0
0 0
1
2
2

Lenght
EXTENDED
500.0
1000.0

Height
EXTENDED
10.0
50.0

1 2
```

You can see the structure of the *test_abaqus.bio*, *test_abaqus.pop* and *test_abaqus.sta* output files in the following paths:

BIANCA 3.1 User guide\ Examples\ test_abaqus.bio;
BIANCA 3.1 User guide\ Examples\ test_abaqus.pop;
BIANCA 3.1 User guide\ Examples\ test_abaqus.sta.

We perform also the post processing of results and we show here below the structure of the *post_processing.inp* input file:

```
MATLAB
linear
off
png
```

In this case we obtain two image files saved as Portable Network Graphics files named *obj_min_test_abaqus.png* and *obj_mean_test_abaqus.png*. You can find those files in the following paths:

BIANCA 3.1 User guide\ Examples\ obj_min_test_abaqus.png;
BIANCA 3.1 User guide\ Examples\ obj_mean_test_abaqus.png.

Figs. 3.15 and 3.16 show the curves of the best feasible solution vs generations

and the average value of the objective function vs generations, respectively.

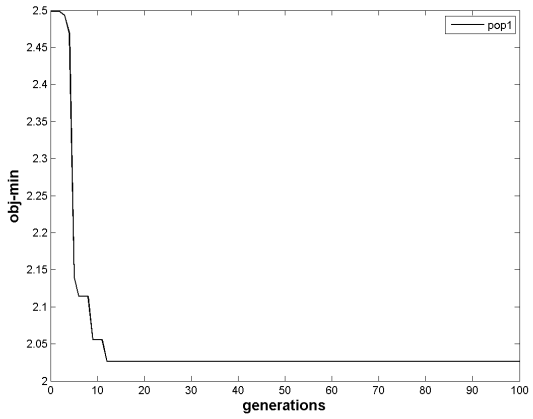

Figure 3.15: Best feasible solution vs generations for the stiffened plate design problem.

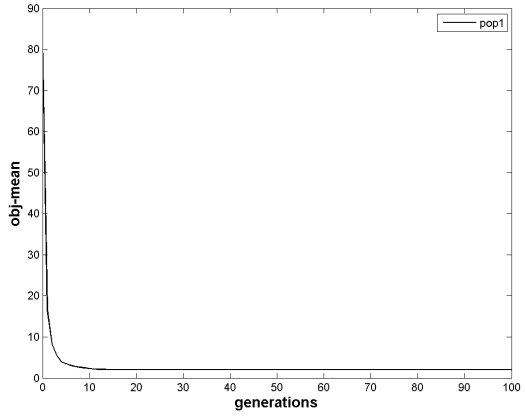

Figure 3.16: Average value of the objective function vs generations for the stiffened plate design problem.

3.7 Example of interface with CAST3M® code: plate design problem

Bibliography

[1] A. Vincenti. *Conception et optimisation de composites par mèthode polaire et algorithmes génétiques*. PhD thesis, Université de Bourgogne, France, 2002.

[2] A. Vincenti, M.R. Ahmadian, and P. Vannucci. Bianca: a genetic algorithm to solve hard combinatorial optimisation problems in engineering. *Journal of Global Optimisation*, 48:399–421, 2010.

[3] M. Montemurro, A. Vincenti, P. Vannucci, and A. Makradi. Design of laminate's elastic properties with minimum number of plies. *Composites Part A*, (Submitted), 2010.

[4] M. Montemurro, P. Vannucci, and A. Vincenti. A genetic algorithm with cross-over on species for structural optimisation. *Computers and Structures*, (Submitted), 2010.

[5] D.E. Goldberg. *Genetic algorithms*. New York: Addison and Wesley, 1994.

[6] Z. Michalewicz. *Genetic algorithms + data structures = evolutionary programming*. Berlin: Springer, 1994.

[7] P. Vannucci. Designing the elastic properties of laminates as an optimisation problem: a unified approach based on polar tensor invariants. *Structural and Multidisciplinary Optimisation*, 31(5):378–387, 2006.