Optimisation Techniques for Industrial Spring Design

Manuel Paredes, Marc Sartor, Jean-Christophe Wahl

OPTIMISATION TECHNIQUES FOR INDUSTRIAL SPRING DESIGN

Manuel PAREDES, Marc SARTOR, Jean-Christophe WAHL
Dpt Génie Mécanique, INSA, 135 avenue de Rangueil 31077 Toulouse Cedex 4
manuel.paredes@insa-toulouse.fr

Abstract

The main industrial software available to a designer during a spring definition work use exhaustive calculations from standards to make a full check of the proposed spring's compatibility with the given specifications. As far as we are aware, they don't really exploit optimisation process capabilities. This paper presents tools that exploit several optimisation techniques for custom spring design by linking both industrial and mathematical knowledge.

We have defined tools that manage not only design parameters but also operating parameters including designer specifications, calculation from standards (buckling, fatigue life, operating limits) and the capability limits of the spring manufacturer.

Each tool uses a specification sheet where data is set with interval values. Setting data in this way provides a powerful and efficient means of expression for designers during the early stages of the design process. The tools we have developed for custom spring design are specific for each type of springs.

The compression spring optimisation implies solving an optimisation problem with 6 continuous variables (4 design variables and 2 operating variables) while satisfying 43 constraints. The resolution process is based on a mathematical programming process. An algorithm to automatically intialise the variables has been developed.

The extension spring optimisation tool is based on the same process. In that case, when a specific angle between end loops is required, the optimisation problem is defined by 5 continuous variables, one integer variable (the number of coils) and 43 constraints. The algorithm uses the previous resolution method for continuous variables and adds a branch and bound process in order to find the mixed variables optimum.

The torsion spring optimisation has required the use of a hybrid resolution algorithm. The optimisation problem is defined by 7 continuous variables (3 design variables and 4 operating variables) and 45 constraints. In that case the constraints can define a non convex allowable space for the variables. For that kind of

problems, deterministic mathematical programming fails in jumping over the several intervals. For that reason, we have implemented a process that first uses an evolutionary strategy. The best solution found is then improved by a mathematical programming process.

To deal with multi-objective optimisation, we have chosen to use an interactive dialog with the designer : each possible objective function can be set as a constraint in the specification sheet. That way, the designer can try several configurations and find the best Pareto optimal solution for his application.

All the presented tools have been implemented on visual-basic for Excel. They have been successfully tested by a spring manufacturer on industrial problems. All this study shows the efficiency and the benefits that can be obtained on industrial applications using design tools that exploit both deterministic and stochastic optimisation processes.

1. Introduction

The design of machines imposes the dimensioning of numbers of common mechanical components (gears, cams, shafts, springs). These components have their own dimensioning rules and require specific manufacturing knowledge. The problem of designing a mechanical component is often solved by using tables and charts for certain pre-selected specifications and certain pre-selected objectives. These calculations can be carried out manually, but without computer assistance, designers are often obliged to oversimplify the procedures, *e.g.* by assigning a value to certain parameters in order to reduce the number of problem variables to only 2 or 3 [1,2]. So, they are unable to exploit all the specification possibilities and consequently to optimize design.

Progress in computer assisted design should lead to successful solutions to this kind of problem. In theory, it should be possible, for a given component, to define the problem globally and find the appropriate resolution method that would provide the optimal solution whatever the specifications. This paper focuses on the optimum design of helical springs with circular wire [3] which are the most common springs in use today.

Spring design has been often used to illustrate optimisation algorithms. Yokota, Taguchi and Gen [4] present an optimum spring design using genetic algorithms [5]. Deb and Goyal [6], Kannan and Kramer [7] and Sandgren [8] compare the efficiency of their optimisation algorithms using the same example dealing of spring design. Unfortunately, all the examples are very simplified and can not be directly used by designers in industry.

From another point of view, the main industrial software available to a designer during the spring definition work (from IST [9], from Hexagon [10] or from

SMI [11]) use standard calculations only to make a full check of the proposed spring's compatibility with the given specifications. "Spring Design Software" from SMI can carry out minimum mass optimisation to adjust one design parameter but three other design parameters have to be known beforehand. Industrial software dedicated for spring design can be considered as robust validation software. They are thus mainly used at the very end of the design cycle, when the detailed design is considered.

There is a need for tools that could be used at any steps of the design cycle, from the very beginning where most of the data are not known or uncertain, to the detailed design where data are quite well defined. This paper proposes an approach which intends to satisfy this need.

2. Requirements for the tools

It is difficult to propose optimisation tools to industry. Most of designers did not learn optimisation at school and thus do not like using optimisation techniques. For that reson, we have chosen to developp our tools using Excel and visual-basic. These well known software are commonly used in industry. This approach can be a good way to introduce optimisation knowledge to designers.

Considering the expected wide range of applications for the tools, we have decided that each specification sheet should allow the designer to give information upon each design parameters and each operating parameters. In a specification sheet, each parameter can be defined by a lower limit and/or an upper limt as shown in figure 1. The idea is to let the designer fullfiled the knowned cells without taking care of the empty ones.
In order to be able to defined the best design, an objective function as to be selected in the specification sheet.

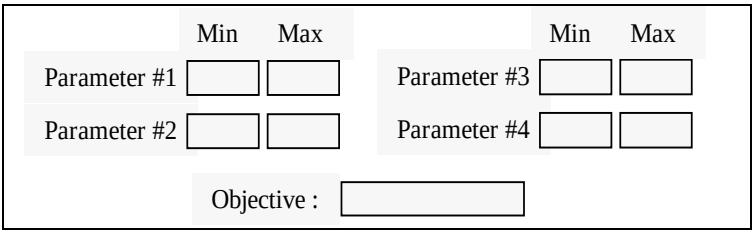

Figure 1. Proposed specification sheet

3. Automatically build the optimisation problem

Once the requirements are defined, the results have to be automatically calculated. The first step consists on building the problem associated to the specifications.

The problem can be expressed as an optimisation problem :

Minimise or Maximise F(X)
Subjected to constraints

The objective function F is selected by the designer in the specification sheet. It can be any parameter that is considered in the specification sheet.
The vector of variables X is chosen in order to define the spring (design variables) and the associated use of the spring (operating variables).
The constraints are defined not only by the limits defined on the specification sheet but also by the manufacturing constraints and the requirements from standards.
Depending on the considered type of spring, on the selected objective function, on the fullfiled cells and on the associated values, very differents optimisation problems can be obtained. In any case, the resolution process has to automatically find a solution without any assistance from the designer. Considering this, we have defined resolution methods dedicated for each type of spring.

4. Automatically solve the problem for compression springs

Figure 2 presents the main parameters defining helical compression springs. All the parameters that can be used by a designer to define and use a spring appear in the specification sheet presented in figure 3. The objective function selected in the specification sheet, the requirements defined by the designer, the requirements from standards and from the manufacturers define an optimisation problem with 6 continuous variables (4 design variables and 2 operating variables) while satisfying 43 constraints.

Considering that we have to manage continuous variables and convex spaces, we have sought that we could take benefits of mathematical programming [12] algorithms. We have selected a direct method (GRG) as it is able to provide an acceptable solution, if non optimal, even if the resolution process is stopped before end (for example if the resolution process is too long). The GRG method is already implemented in the Excel solver.

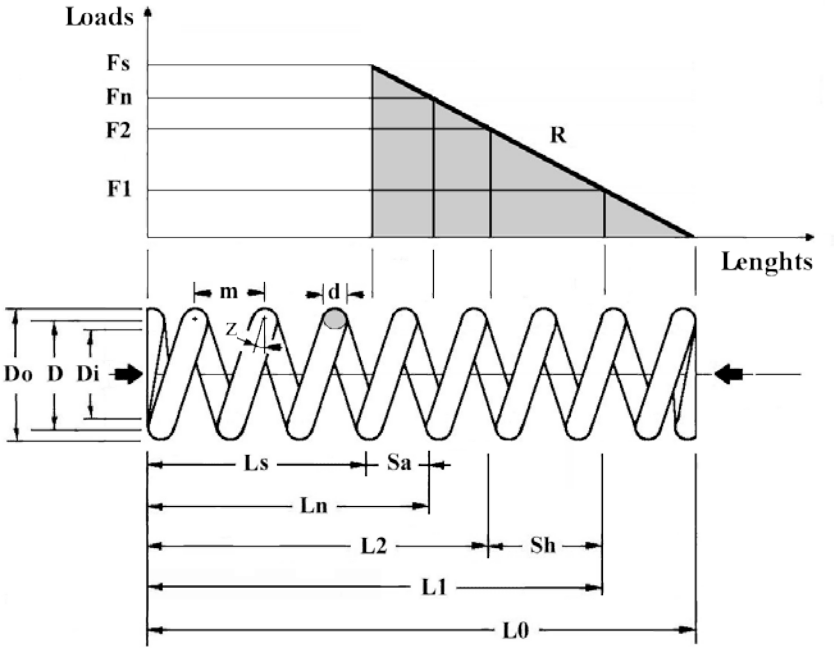

Figure 2. Compression spring parameters

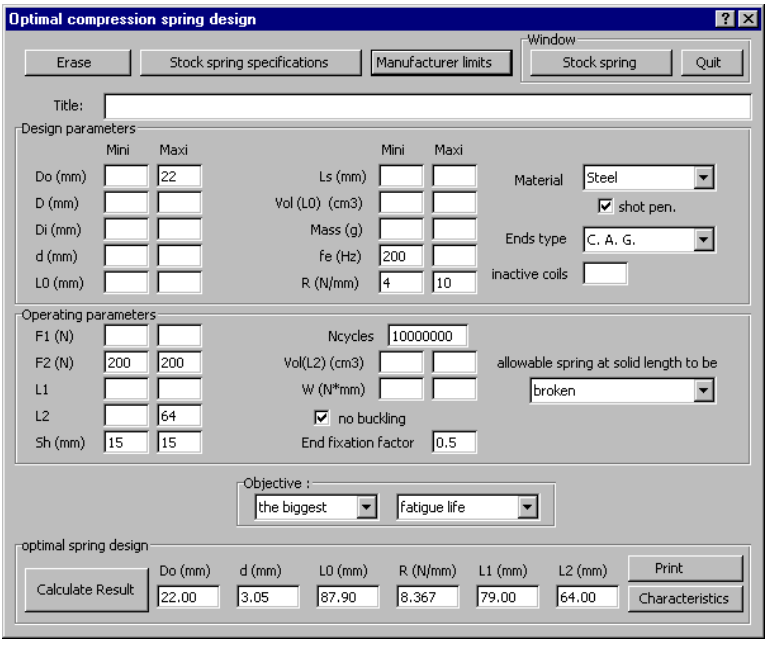

Figure 3. Specification sheet for compression springs

Direct methods need the variables to be well initialised *i.e.*, that the calculation starting point is an acceptable solution. For that reason, we have developed another algorithm to initialise the variables. It uses interval arithmetic [13] to perform a sequential meshing of the space of the design variables in order to build a virtual catalogue related to the specifications. The best spring form this virtual catalogue is used to initialise the variables for the mathematical programming process.

To build the virtual catalogue, limits from standards, manufacturer constraints and the most usual data (*Do*, *D*, *Di*, *d*, *L0*, *R*, *F1*, *F2*, *L1*, *L2*, *Sh*) from the specification sheet are exploited using interval arithmetic in order to define springs which respect the majority of the problem constraints.

Interval arithmetic is thus exploited to reduce the variables limits depending on the specifications. To build the virtual catalogue, a basic method could consist on first determining all variable limits depending on the specifications and on secondly regularly meshing the variable space (each node representing a spring of the virtual catalogue). In fact, sequential meshing is a more efficient method. The idea is to recalculate variable limits before meshing a one variable space. Once a first regular meshing (on one variable is performed), another meshing can be performed on another variable for each fixed value of the preceding variable depending on those values.

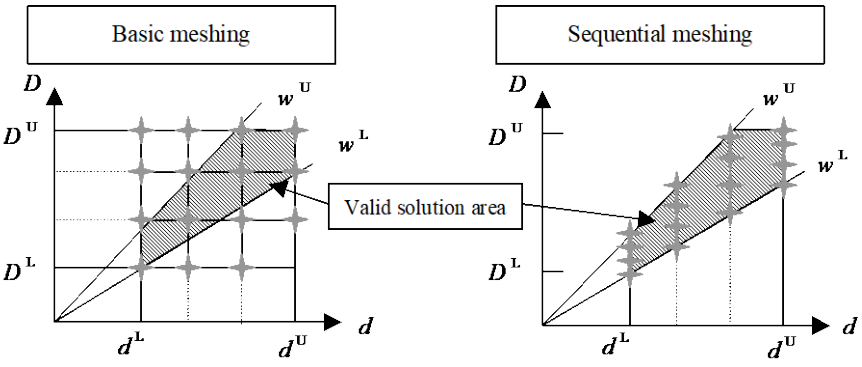

Figure 4. Comparison on two meshing methods

An example of the difference between a basic and a sequential meshing is illustrated on figure 4. This example uses a simplified problem which involves two variables (*d* and *D*) and only 6 constraints representing *d*, *D* and *w* (*w*=*D*/*d*) limits. Using the basic meshing method, the allowable variation space for the two variables are first calculated and then the obtained space is regularly meshed. Using the sequential meshing, the algorithm starts meshing the available variation space for the first variable *d*. Secondly, for each obtained value of *d*, the algorithm recalculates the available limits for the second variable *D* and meshes the associated spaces. In this example, a virtual catalogue containing 16 springs is

built. Using a basic meshing, only 6 springs fit the specifications whereas using a sequential meshing, all the springs respect the specifications. That is the reason why we use a sequential meshing of the variables space to build the virtual catalogue.

Once the virtual catalogue is built, the algorithm developed by Paredes et al. [14] is used to select the best stock spring from the virtual catalogue and determine the associated operating lengths.

We have called the proposed resolution process SM + MP (Sequential Meshing and Mathematical Programming).

In the example presented in figure 3, the SM process has built a virtual catalogue containing 1256 springs. From within this catalogue, 262 springs fit the specifications. The best spring has been selected to initialise the variables for the MP process. The evolution of the objective function value during the MP process is presented on figure 5. The GRG algorithm has quickly converged towards the optimal solution. The overall process is no longer than 6 seconds on a 300Mhz computer.

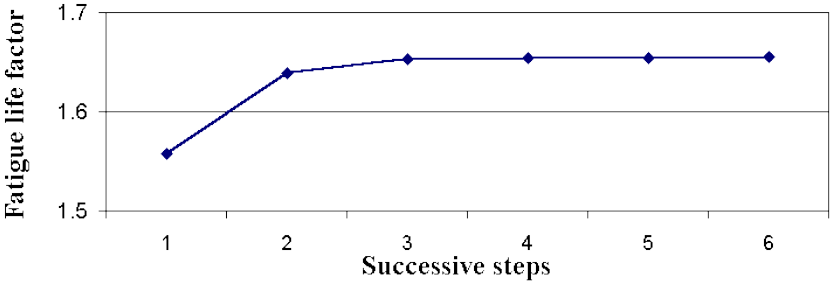

Figure 5. Evolution of the objective function during the MP process

These results illustrate the efficiency of the proposed SM + MP process.

5. Automatically solve the problem for extension springs

The main parameters that can be used to defined extension springs are presented on figure 6. The extension spring optimisation implies solving an optimisation problem with 6 variables (4 design variables and 2 operating variables) while satisfying 43 constraints.

Most of the time, the variables can be considered as continuous variables thus the SM + MP process presented before is used.

When a designer needs a fixed angle between end loops, the number of coils *n* becomes a discrete variable. To solve this mixed variable problem, a branch and bound process is added to the first resolution process. The branch and bound process is already implemented the Excel solver.

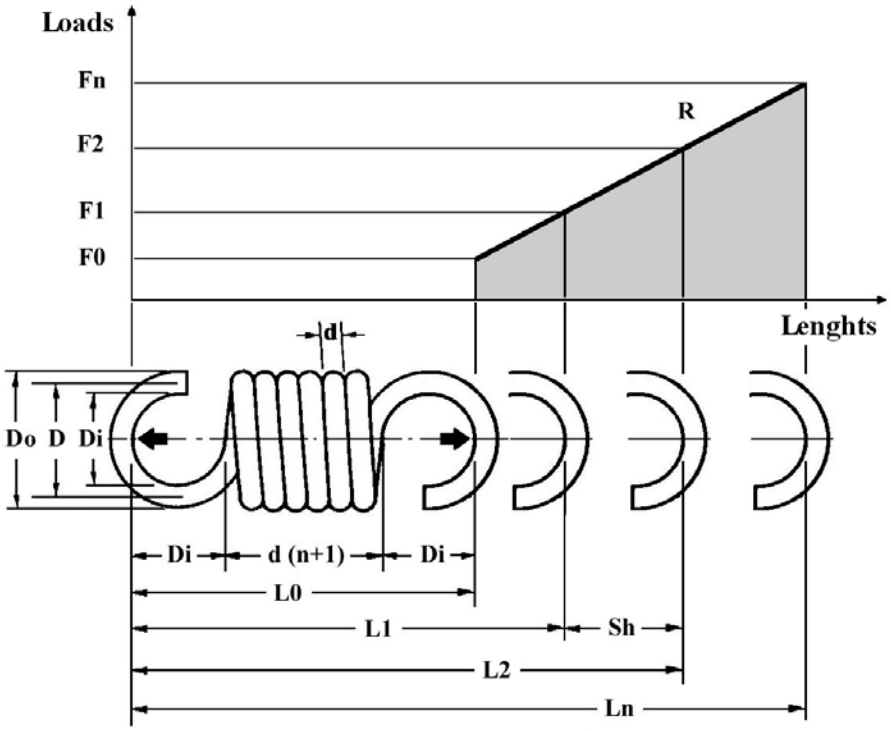

Figure 6. Extension spring parameters

6. Automatically solve the problem for torsion springs

Torsion springs differ from both compression and extension springs in both load application and mode of operation. Helical torsion springs are often used in non highly stressed applications. This is mainly due to the fact that there are few fatigue life data available as friction problems appear between the coils and at the end of the legs where the load is applied. The shape of legs is usually determined by the required mode of operation of the component to which it is attached. However, designers should always try to use the simplest legs *i.e.*, torsion springs with tangential legs. Design and operating parameters of torsion springs with tangential legs are presented in figure 7.

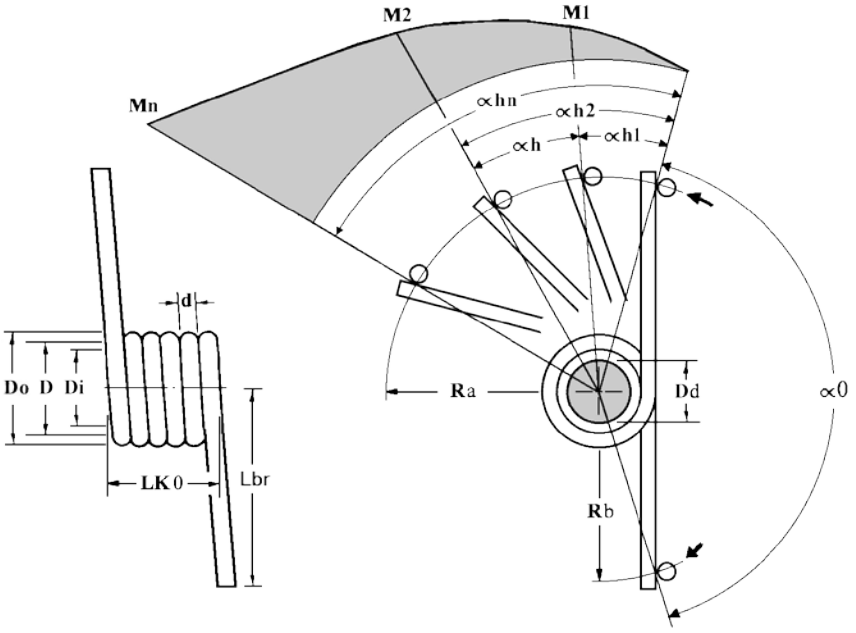

Figure 7. Torsion springs parameters

Even considering static applications, custom torsion spring design is more complicated that the design of compression or extension springs. The torsion spring optimisation has required the use of a hybrid resolution algorithm.

The optimisation problem can be defined by 7 continuous variables (3 design variables and 4 operating variables) and 45 constraints. In some cases, constraints can define a non convex allowable space for the variables.

An example is presented on figure 8. Specifications on the angle between ends such as $90° < \gamma < 180°$ implies a number of coils defined by intervals : $N + 0.25 < n < N + 0.5$ where N is an integer value. For that kind of problems, deterministic mathematical programming fails in jumping over the several intervals. For that reason, we have implemented a process that first uses an evolutionary strategy (ES). The idea is that ES is able to find the area where the optimum solution is located. The best solution found by ES is then improved by the MP process previously described.

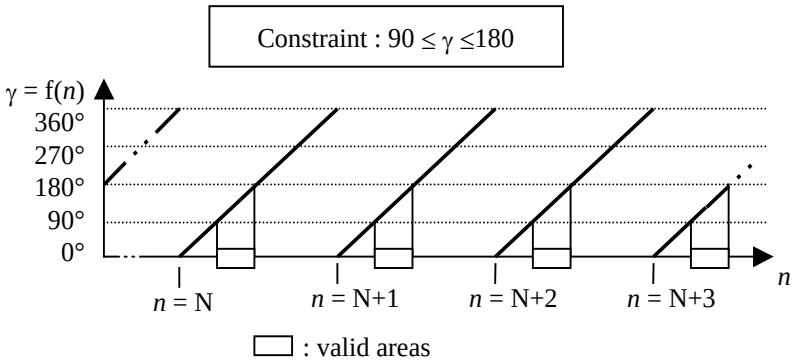

Figure 8. The number of coils can be defined on a non convex space

7. Multi-objective optimisation

There are many way to deal with multi-objective optimisation [15]. We have chosen to use an interactive dialog with the designer : a constraint can be set in the specification sheet on each possible objective function. That way, the designer can try several configurations and find the best Pareto optimal solution for his application as described in figure 9.

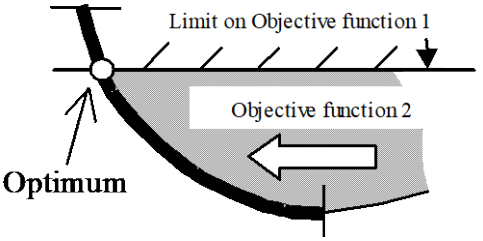

Figure 9. Find the best solution on the Pareto front

8. Conclusion

We have shown that defining the best custom spring for an application could be set as an optimisation problem. Tools for compression, extension and torsion spring design have been defined. Several resolution methods depending on the considered type of springs have been implemented. They use interval arithmetic, a direct optimisation method, branch and bound process and an evolutionary strategy.

The proposed tools dedicated to optimal spring design have been implemented on visual-basic and Excel to be easily used in industry. They have been successfully tested by a spring manufacturer on industrial problems.

The short resolution and preparing time of the proposed tools allows designers to tests several configurations. They thus can use them as exploration tools by adding data, refining data or changing the objective function from a problem to another. This type of tool increases design process flexibility as the component design is made easier and more efficient.

All this study shows the efficiency and the benefits that can be obtained on industrial applications using design tools that exploit both deterministic and stochastic optimisation processes.

Acknowledgements

The financial and technical support of the spring manufacturer « Ressorts VANEL » is gratefully acknowledged.

References

1. Kothari H (1980) Optimum design of helical springs. Machine design, 69-73
2. Metwalli S, Radwan A and Elmeligy A A (1994) CAD and Optimization of helical torsion springs. ASME Computers in Engineering 767-773
3. WAHL A M (196) Mechanical Springs. McGraw-Hill, New York
4. Yokota T, Taguchi T, Gen M, (1997) A Solution Method for optimal Weight Design Problem of Herical Spring Using Genetic Algorithm. Elsevier, Computers ind. Engng. 33:71-76
5. Davis L, (1991) Handbook of genetic algorithms. International Thomson computer press
6. DEB K and GOYAL M (1998) A Flexible Optimization Procedure for Mechanical Component Design Based on Genetic Adaptive Search. ASME Journal of Mechanical Design 120 : 162-164.
7. KANNAN B K and KRAMER S N (1994) An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design. ASME Journal of Mechanical Design 116 : 405-411.
8. SANDGREN E (1990) Nonlinear Integer and Discrete Programming in Mechanical Design Optimization. ASME Journal of Mechanical Design 112 : 223-229.

9. IST, Institute of Spring Technology, Henry Street Sheffield S3 TEQ United Kingdom
10. Hexagon, GmbH, Stiegelstrasse 8, D-73230 Kirchheim/Teck, Deutschland
11. SMI, Spring Manufacturer Institute, 2001 Midwest Road, Suite 106, Oak Brook, Illinois 60523-1335, USA
12. Vanderplaats G N (1984) Numerical optimization Techniques for Engineering Design. Mac Graw Hill
13. Moore R E (1979) Methods and applications of interval analysis. SIAM Studies in Applied Mathematics
14. Paredes M, Sartor M, Fauroux J-C (2000) Stock spring selection tool. Springs, official publication of the Spring Manufacturer Institute 39: 53-67
15. Deb K (2000) Multi-Objective Evolutionary Optimization : Past, Present, and Future. Springer, Evolutionnary Design and Manufacture, Selected Papers from ACDM ’00, pp.225-236