# A Multi-Objective Optimization with Open Source Software

Sometimes it happens that a small-to-medium sized firm does not benefit from the advantages that could be achieved through the use of virtual simulation and optimization techniques. This represents in many cases a great limitation in the innovation of products and processes, and this can lead, in a very short time, to a complete exclusion from the market and to an inevitable end.
Nowadays, it is mandatory to reduce as much as possible the time-to-market, while always improving the quality of products and satisfying the customer needs better that the competitors. In some fields it is a question of "life or death".

According to our opinion, the main reasons that limit or, in the worst case, make impossible the use of the virtual simulation and optimization techniques can be grouped into three categories:

1. These techniques are not yet sufficiently known and the possible users do not have a great confidence in the results. Sometimes physical experimentation, guided by experience maturated through many years of practice, is thought to be the only possible way to proceed. This is actually wrong in the great majority of cases, especially when new problems have to be solved. A change of vision is the most difficult but essential step to take in this context.

| | Rough Phase | Fine Phase |
|---|---|---|
| License | Many possibilities are available | GNU license largely used or similar versions with some restrictions |
| Development | Continuous improvement and a clear guideline | Left to the community |
| Available features | State of the art | It strongly depends on "who" leads the development. Sometimes, very advanced features can be available. |
| Technical support | Usually the distributor offers a technical support | Usually no support is available but in some cases forums can help |
| Usability | Easy-to-use and smart GUIs | Some effort could be required to the user |
| Customization | Only in some cases | If the source code is available the possibility of customization and development is complete |

*Table 1: The table compares some key features that characterize commercial and open source software, according to our opinion.*

2. Adequate hardware facilities considered necessary to perform an optimization are not available and therefore the design time becomes too long. We are convinced that, in many cases, common personal computers are enough to efficiently solve a large variety of engineering problems. So, this second point, which is often seen as an enormous obstacle, has to be considerably downsized.
3. The simulation software licenses are much too expensive given the firm's financial resources. Even though the large majority of commercial software houses offer a low-cost first entry license, it is not always immediately evident that these technologies are not just an expense, but rather a good investment.

As briefly stated above, the second point often does not represent a real problem; the most important obstacle is summarized in the first point. People actually find it hard to leave a well-established procedure, even if obsolete, for a new one which requires a change in the everyday way of working. The problem listed in the third point can be solved, when possible, by using open source (see [1]), free and also home-made software. It is possible to find, with an accurate search on internet, many simulation software systems which are freely distributed by the authors (under GNU license in many cases). Some of them also exhibit significant features that usually are thought to be exclusive to commercial software.
As usual, when adopting a new technology, it is recommended to consider both the advantages and the disadvantages. We have compared in Table 1 some aspects that characterize the commercial and the open source codes which should be considered before adopting a new technology.
Open source codes are usually developed and maintained by researchers; contributions are also provided by advanced users all over the world or by people who are supported by research projects or public institutions, such as universities or research centers. Unfortunately, this can lead to a discontinuous improvement, not driven by a clear guideline, but rather left to the free contributions given by the community. On the contrary, commercial software houses drive the development according to well-known roadmaps which generally reflect specific industry trends and needs.
Commercial software is usually "plug-and-play": the user has just to install the package and start using it. On the contrary - but not always - open source software could


Case Studies

require some skill and effort in compiling the code or adapting a package to a specific system configuration.
Software houses usually offer to the customer technical support, which can be, in some cases, really helpful to make the investment profitable. An internet forum, when it exists, is the only way to have support for a user of an open source code.
Another important issue is the usability of the simulation software, which is mainly given by a user-friendly graphical interface (often referred to as GUI). The commercial software usually has sophisticated graphical tools that allow the user to easily manage and explore large models in an easy and smart way; the open source codes rarely offer a similar suite of tools, but they have simpler and less easy-to-use graphical interfaces.
The different magnitude of the investment can explain all these differences between the commercial and open source codes.

*Fig. 1 - An example of C-type press. The steel C-shaped profile which will be optimized in this work is highlighted with a red line.*

However, there are some issues that can make the use of a free software absolutely profitable, even in an industrial context. Firstly, no license is needed to run simulations: in other words, no money is needed to access the virtual simulation world. Secondly, the use of open source software allows to break all the undesired links with third party software houses and their destiny. Third, the number of simultaneous runs that can be launched is not limited, and this could be extremely important when performing an optimization process. Last, but not least, if the source code is available all sorts of customizations are in principle possible.
The results of a structural optimization, performed using only open source software, are presented in this paper. We decided to use Scilab (see [2]) as the main platform to drive the optimization process through its genetic algorithm toolbox. For the solution of the structural problem, presented in the following, we adopted two packages. The first one is the Gmsh (see [3]) to manage a parametric geometry and mesh it; the second one is CalculiX (see [4]), an FEM solver. It is important to remember that this choice is absolutely not mandatory, but is strictly a matter of taste.

| Plate thickness [mm] | Plate max dimensions [m] | Available steel codes |
|---|---|---|
| 30 | Vertical <4<br>Horizontal <2 | A, B, C |
| 50 | Vertical <3<br>Horizontal <2 | A, B |

*Table 2: The table collects some limitations in the steel plate provision.*

| Steel code | Young modulus [MPa] | Maximum stress / Yield limit [MPa] | Cost [$/Kg] |
|---|---|---|---|
| A | 210000 | 200 (220) | 1.2 |
| B | | 300 (330) | 2.3 |
| C | | 600 (630) | 4.0 |

*Table 3: The maximum desired stress, the yield limit and the cost per unit weight for the three available steels.*

**The structural optimization problem**

In this work a C-type press is considered, as the one shown in Figure 2. This kind of geometry is preferred to other ones when the force that has to be expressed by the hydraulic cylinder is not very high, usually not greater than roughly 200 [Ton]. The main advantages of this type of press are essentially the relative low weight and volume of the machine and the possibility of accessing the workbench from three sides.
The dimensioning of the lateral C-shaped profiles is probably one of the most challenging phases in the design process; the large majority of the weight and cost, for the mechanical part at least, is actually concentrated there. Consequently, a good designer looks for the lightest profile which is able to satisfy all the structural requirments. Moreover, an economical configuration is also desired, in order to reduce as much as possible the production cost.

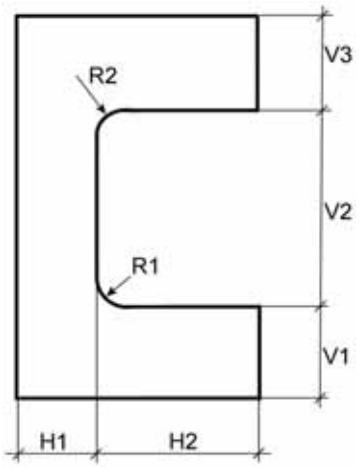


*Fig. 2 - The C-shaped plate geometry has been modeled using the dimensioning drawn in this picture, together with the thickness TH of the plate.*

When optimizing, the designer should also take into account some aspects which are not strictly related to structural issues but are however important or, in some cases fundamental, to deal with an optimal design. These aspects could be related to the availability of materials and components on the market, technical norms that have to be satisfied, marketing indications and more. In our case the steel plate supplier, for example, can provide only the configurations collected in Table 2.
It is clear that an optimization process that does not take into consideration these requisites could lead to configurations which are optimal only from a theoretical point of view, but which cannot be practically implemented. For example, a very light configuration is


Case Studies

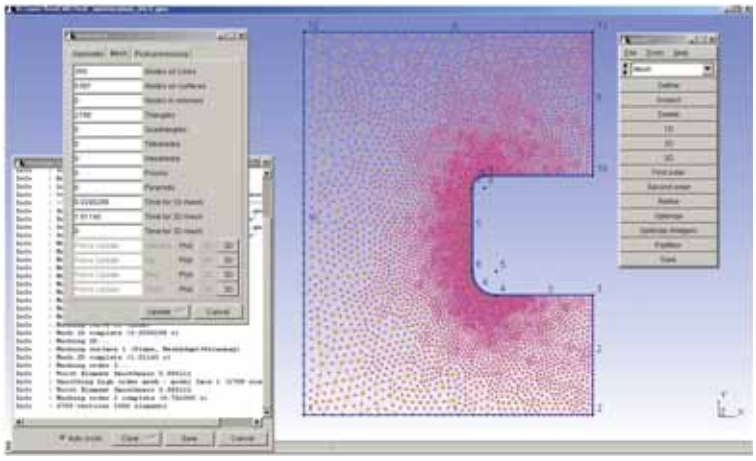

*Fig. 3 - A possible version of the C-shaped plate meshed in Gmsh.*

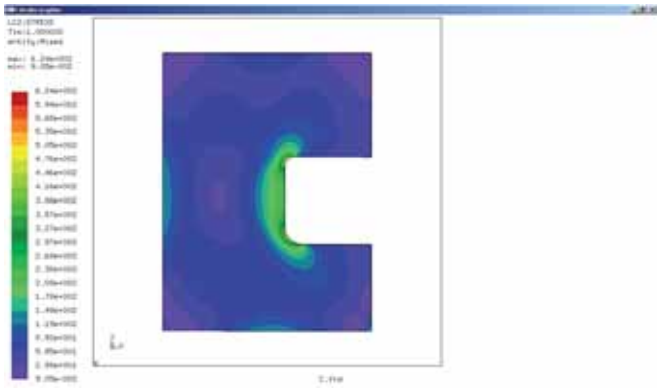

*Fig. 4 - The CalculiX GraphiX window, where the von Mises stress for the C-shaped plate is plotted.*

not of practical interest, if it requires a steel characterized by a yield limit greater that 600 [MPa].
For this reason all the requirements collected in Tables 2 and 3 have been included in order to drive the optimization algorithm to feasible and interesting solutions.

Moreover, it is required that the hollow in the plate (H2-max(R1,R2) x V2, see Figure 2) is at least 500 x 500 [mm] to allow the positioning of the transversal plates and the hydraulic cylinder.

Another technical requisite is that the maximum vertical displacement is less than 5 [mm] to avoid excessive misalignments between the cylinder axis and the structure under the usual operating conditions. This limit has been chosen arbitrarily, in the attempt to exclude the designs that are not sufficiently stiff, taking into account, however, that the C-plate is a part of a more complex real structure which will be much more stiff than what is calculated with this simple model.
A designer should recognize that the solution of such a problem is not exactly trivial. Firstly, it is not easy to find a configuration which is able to satisfy all the requisites listed above; secondly, it is rather challenging to obtain a design that minimizes both the volume of the plate (the weight) and the production cost.

**The open source software for the structural analysis**

Gmsh has been used as a preprocessor to manage the parametric geometry of the C-shaped plate and mesh it in batch mode. Gmsh has the ability to mesh a non-regular geometry using triangular elements; many controls are available to efficiently define the typical element dimension, the refinement depth and more. It is a very powerful software tool which is also able to manage complicated three-dimensional geometries and efficiently mesh them using a rich element library.
The mesh can be exported in a formatted text file where the nodes and the element connectivities are listed together with some useful information related to the so-called physical entities, previously defined by the user; this information can be used to correctly apply, for example, boundary conditions, domain properties and loads to a future finite element model.
The CalculiX finite element software has been used to solve the linear elastic problem. Also in this case a batch run is available; among the many interesting features that this software offers are the easy input text format, and the ability to perform both linear and non-linear static and dynamic analyses.
CalculiX also offers a pre and post processing environment, called CalculiX GraphiX, which can be used to prepare quite complicated models and, above all, display results.
These two software tools are both well documented and also some useful examples are provided for new users. The input and output formats are, in both cases, easy to understand and manage.
In order to make the use of these tools completely automatic, it is necessary to write a procedure that translates the mesh output file produced by Gmsh into an input file readable by CalculiX. This translation is a relatively simple operation and it can be performed without a great effort using a variety of programming languages; a text file has to be read, some information has to be captured and then rewritten into a text file using some simple rules. For this, a simple executable file (named translate.exe) has been compiled and it will be launched whenever necessary.
A similar operation has also to be performed in an optimization context to extract the interesting quantities from the CalculiX result file and rewrite them into a compact and accessible text file.
As before, an executable file (named read.exe) has been produced to read the .dat results file and write the volume, the maximum vertical displacement and the nodal von Mises stress corresponding to a given design into a file named results.out.
Many other open source software codes are available, both for the model setup and for its solution. Also for the results visualization, there are many free tools with powerful features. For this reason the interested reader


Case Studies

can imagine the use of other tools to solve this problem in an efficient way.

### The optimization process driven by Scilab

The genetic algorithm toolbox, by Yan Collette, is available in the standard installation of Scilab and it can be used to solve the multi-objective optimization problem described above. This toolbox is composed of some routines which implement both a MOGA and a NSGA2 algorithm and also a version for the operations that have been performed when running a genetic algorithm, that is the encoding, the crossover, the mutation and the selection.

| Variable | Lower bound [mm] | Upper bound [mm] | Step [mm] |
|---|---|---|---|
| H1 | 250 | 150. | 5 |
| H2 | 500 | 1500 | 5 |
| V1 | 250 | 1500 | 5 |
| V2 | 500 | 1500 | 5 |
| V3 | 250 | 1500 | 5 |
| R1 | 50 | 225 | 5 |
| R2 | 50 | 225 | 5 |
| TH | 20 | 50 | 10 |

*Table 4: The lower and upper bounds together with the step for the input variables.*

These routines are extremely flexible and they can be modified by the user according to his or her own needs, since the source code is available. This is exactly what we have done, modifying the optim_moga.sci script to handle the constraints (with a penalty approach) and manage the infeasible designs efficiently (i.e.: all the configurations which cannot be computed); we have then redefined the coding_ga_binary.sci to allow the discretization of the input variables as listed in Table 4. Other small changes have been made to the routines to perform some marginal operations, such as writing partial results to a file.

When the genetic algorithm requires the evaluation of a given configuration, we run a Scilab script which is charged to prepare all the text files needed to perform the run and then launch the software (Gmsh, CalculiX and the other executables) through a call to the system in the right order. The script finally loads the results needed by the optimization.

| Design name | H1 [mm] | H2 [mm] | V1 [mm] | V2 [mm] | V3 [mm] | R1 [mm] | R2 [mm] | TH [mm] | Cost [$] | max vM stress [MPa] | max vertical displacement [mm] | Volume [$mm^3$] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 670 | 665 | 575 | 500 | 490 | 165 | 110 | 20 | 1304 | 577.7 | 4.93 | $3.53 \cdot 10^7$ |
| B | 1155 | 695 | 725 | 545 | 840 | 185 | 165 | 30 | 1097 | 199.8 | 1.73 | $1.06 \cdot 10^8$ |

*Table 5: The optimal solutions.*

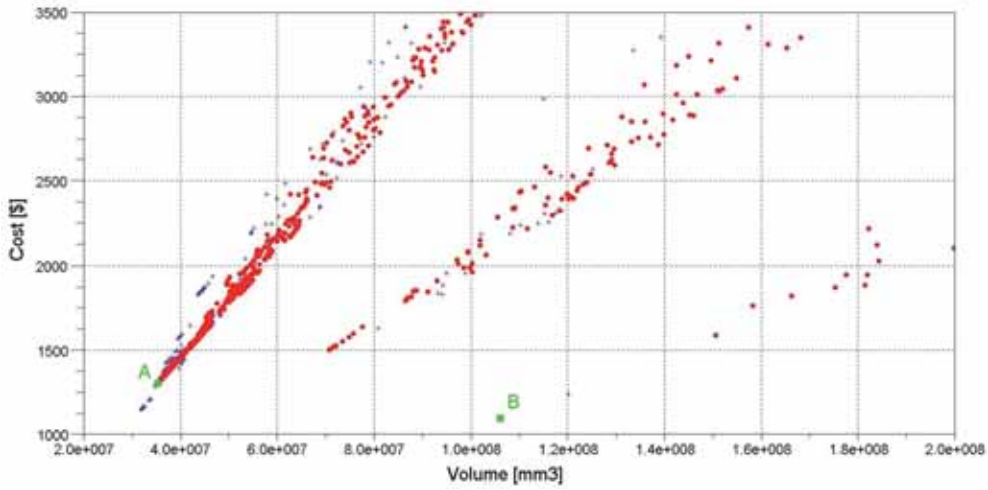


*Fig. 5 - The Cost of the computed configurations can be plotted versus the Volume. Red points stand for the feasible configurations while the blue plus indicates the configurations that do not respect one constraint at least. The two green squares are the Pareto (optimal) solutions (A and B in Table 5).*

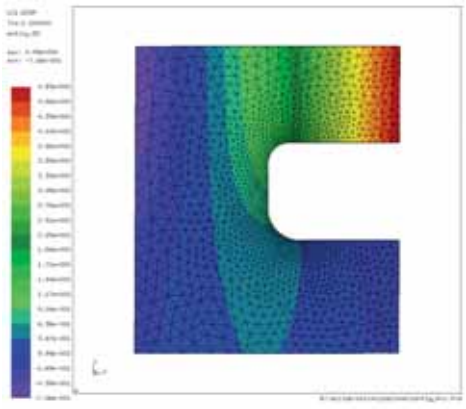

*Fig. 6 - The vertical displacement for the design A.*

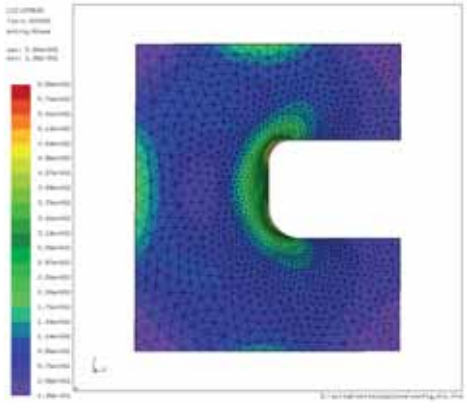

*Fig.7 - The von Mises stress for the design A.*

It is important to highlight that this script can be easily changed to launch other software tools or perform other operations whenever necessary. In our case, eight input variables are sufficient to completely parameterize the geometry of the plate (see Figure 2): the lower and upper bounds together with the steps are collected in Table 4. Note that the lower bound of variable V2 has been set to 500 [mm], in order to satisfy the constraint on the minimal vertical dimension required for the hollow.

We can use a rich initial population, (200 designs randomly generated), considering the fact that a high number of them will violate the imposed constraints, or worse, be unfeasible. The following generations will however consist of only 50 designs, to limit the optimization time.


Case Studies

After 50 generations we obtain the results plotted in Figure 5 and Table 5, where the two Pareto (optimal) solutions are collected. We finally decided to choose, between the two optimal ones, the configuration with the lowest volume (named as "A" in Table 5).

In Figures 6 and 7 the vertical displacement and the von Mises stress are plotted for the optimal configuration named "A" (see Table 5). Note that during the optimization, the maximum value of the von Mises stress computed in the finite element Gauss points has been used, while in Figure 7 the von Mises stress extrapolated by CalculiX to the mesh nodes is plotted; this is the reason why the maximum values are different. However, they are both less than the yield limit corresponding to the steel type C, as reported in Table 3.

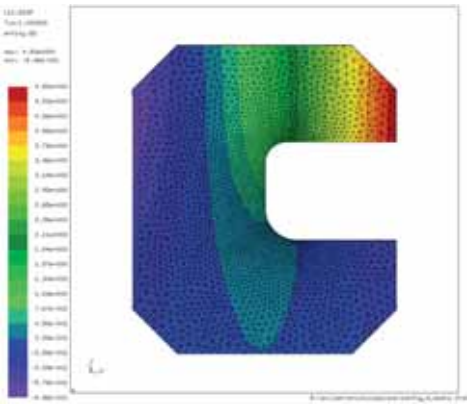

*Fig. 8 - The vertical displacement for the modified design.*

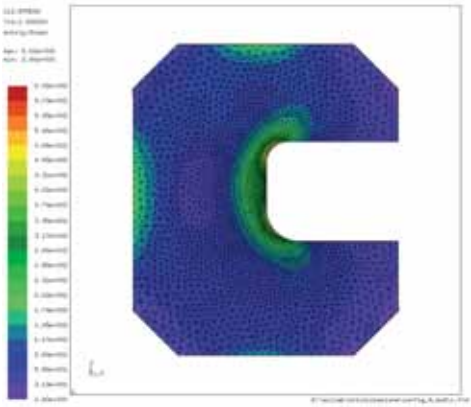

*Fig. 9 - The von Mises stress for the modified design.*

| Horizontal and vertical length of cuts starting from corners [mm] | Cost [$] | max vM stress [MPa] | max Vertical displacement [mm] | Volume [$mm^3$] |
|---|---|---|---|---|
| H1/3 | 1304 | 581.3 | 4.80 | $3.33 \cdot 10^7$ |

*Table 6: The modified design. It can be seen that there is an interesting reduction in the volume with respect to the original design, the "A" configuration in Table 5. Other output quantities do not present significant variations.*

Another interesting consideration is that the Pareto front in our case consists of just two designs: this shows that the solution of this optimization problem is far from trivial.

The design of the C-shaped plate can be further improved. If we run other generations with the optimization algorithm better solutions could probably be found, but we feel that the improvements that might be obtained in this way do not justify additional computations. Substantial improvements can be achieved in another way. Actually, if we look at the von Mises stress distribution drawn in Figure 7 we note that the corners of the plate do not have a very high stress level and that they should not influence the structural behavior very much.

A new design can be tested, cutting the corners of the plate; for the sake of simplicity we decided to use four equal cuts of horizontal and vertical dimensions equal to H1/3, starting from the corners. The results are drawn in Figures 8 and 9, which can be compared with Figures 6 and 7.
As expected, there is a reduction in volume with respect to the original design, but no significant variations are registered in the other outputs. This corroborates the idea that the cut dimensions can be excluded from the set of input variables, since the output does not strongly dependent on them, and this leads to a simpler optimization problem.
The cost does not change; actually it represents the cost of the rectangular plate needed to produce the C-shaped profile.

Other configurations with a lower volume can be probably found with some other runs; however, the reader has to consider that these improvements are not really significant in an industrial context, where, probably, it is much more important to find optimal solutions in a very short time.

**Conclusions**

In this work it has been shown how it is possible to use open source software to solve a non-trivial structural optimization problem.
Some aspects which characterize the commercial and the open source software have been stressed in order to help the reader to make his or her own best choice. We are convinced that there is not a single right solution but rather that the best solution has to be found for each situation.
Whichever the choice, the hope is that virtual simulation and optimization techniques are used to innovate.

**References**

[1] Visit http://www.opensource.org/ to have more information on open source software
[2] Scilab can be freely downloaded from: http://www.scilab.org/
[3] Gmsh can be freely downloaded from: http://www.geuz.org/gmsh/
[4] Calculix can be freely downloaded from: http://www.calculix.de/

**Contacts**

For more information on this document please contact the author:
Massimiliano Margonari - Enginsoft S.p.A.
info@enginsoft.it


Case Studies