PRACTICAL PROBLEMS OF NUMERICAL OPTIMIZATION IN AEROSPACE SCIENCES

Jacek Mieloszyk
Warsaw University of Technology
ul. Nowowiejska 24
00-665 Warsaw, Poland
jmieloszyk@meil.pw.edu.pl

Abstract. Design processes of aircrafts are well established today and described by numerous positions in literature. This well understood and safe path leads to aircraft designs, which become very similar. It is harder then ever to achieve competitive construction. This is why numerical optimization is becoming standard tool during the design process. Although, optimization procedures are becoming more mature, in the industry practice still fairly simple examples of optimization are present. The more complicated is the task to solve the harder it is to implement automated optimization procedures. The paper presents practical examples of optimization in aerospace sciences. Encountered problems during the optimization are presented, solutions of the problems are shown and resulting consequences are discussed.

Keywords. numerical optimization, aircraft design

1 Introduction

Practical numerical optimization in aerospace is most often multidisciplinary. Analysis like aerodynamics, strength, flight dynamics and other are often done by standalone programs [1][2]. Combining few scientific disciplines Figure 1 is always demanding and computationally expensive. Designers and optimization code developers always seek for the way to speed up the computations. The author of the article also has some experience in applied numerical optimization [3][4]. Generally there are few possible ways for improvement: more efficient optimization algorithms (mathematical operations), faster analysis of the objective function by the simulation programs, and reduction of number of the objective function evaluations to reach the optimum. In the process of improving the computational efficiency, quality of the obtained optimum can be accidentally of second importance. This can have significant implications on the final design.

With help of today's widespread multiprocessor computers numerical optimization can be run parallel. Analysis of population of designs, or computation of directional gradient vector are perfectly applicable for the parallelization. This capability is very beneficial, but also complicates, and extends time to define the optimization task.

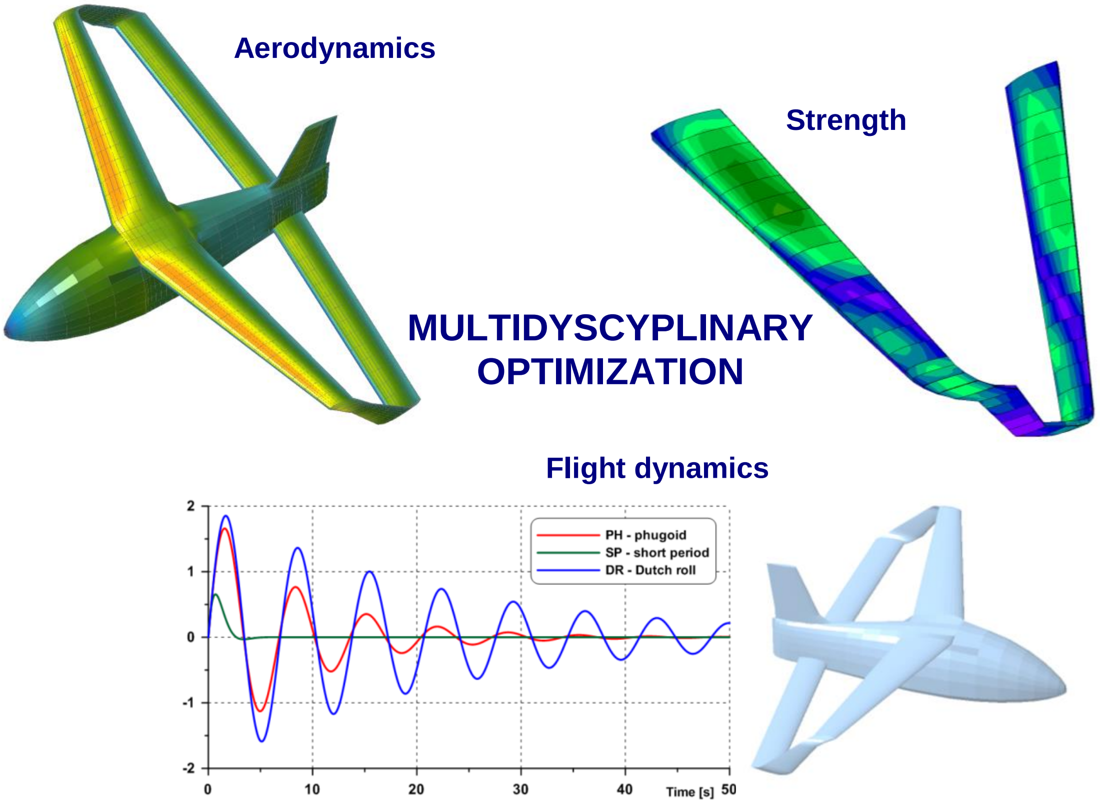

Figure 1: Example of combining many scientific disciplines in optimization.

When the optimization task is set and automatic analysis work well, there is still risk of obtaining unfeasible solutions Figure 2. This is dependent for example on the relationships between parts of the optimized geometry and allowed range of change of the design variables. If the optimization task is simple it is possible to predict all possible collisions, but if the number of design variables is significant, interim errors are almost unavoidable. Adjustments of the settings of the optimization task can be time consuming and expensive, what is especially important for industry users [5].

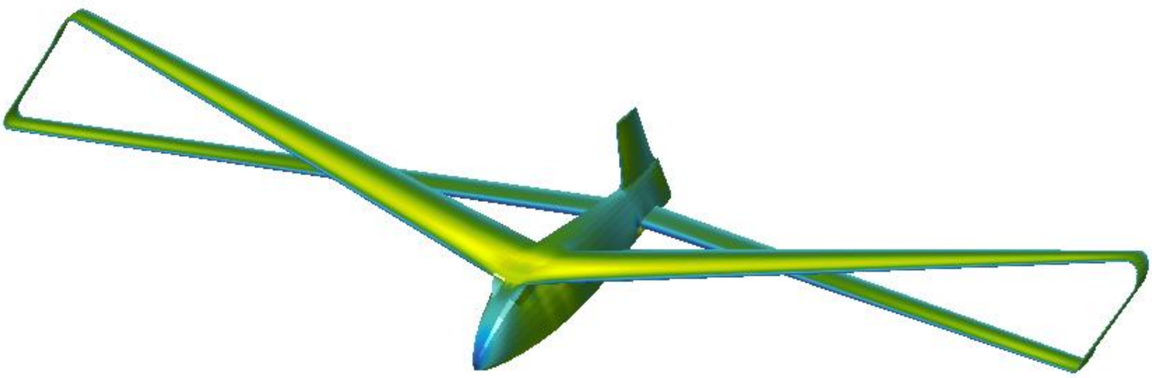

Figure 2: Unfeasible aircraft geometry.

2 Optimization efficiency

There are three major possible places for improvements in optimization procedures, more efficient optimization algorithms, faster analysis software, and improvement of optimization algorithms by reduction of number of objective function analysis. If the software for optimization, and for analysis were created by the same group of scientists there can be achieved improvements on every point. Very often specialists from optimization won’t be the authors of the software for the analyses. If the scientists have access to the optimization code they can improve mathematical operations and redefine optimization algorithms to reduce number of objective function analysis calls. Comparing time of analyses to the time consumed by the optimization software for most practical cases, time for analysis is much longer. In such circumstances the biggest effort should be put on optimization procedures to reduce number of analysis. In the race to achieve the best computational times it is easy to forget about the importance of the optimum quality. Designers have to also pay attention how close the algorithm will get to the desired optimum.

2.1 Quick convergence vs quality of optimum

It can be observed that aggressive algorithms, which might change optimization variables significantly, converge to the optimum quickly. This behavior is not a problem for purely mathematical functions, where variables are defined in the domain of infinite real numbers with high computer accuracy. Situation changes when complex real world problems are considered. The aggressive behavior of the algorithms can cause problems with convergence of the analyses. For example, creating structures, that are too far from the feasible conditions, like negative length of physical dimensions. After overshooting the algorithm will produce wrong solutions, and might not be able to proceed further with the optimization to satisfy the constrains.

The second problem is the behavior of the solution obtained in the neighborhood of the optimum. Numerical model shows predicted value of the objective function in the exactly defined point. The real world products have tolerances of manufacturing, which might have slightly shifted design parameters values. The issue is related to the reliability based optimization [6]. It might happen that the performance of the designed object highly departs from the optimum performance, still being in the prescribed tolerances. The relation is shown on a schematic Figure 3. This can be for example the case of an optimized airfoil and the design parameter that will control very sensitive place of the boundary layer separation.

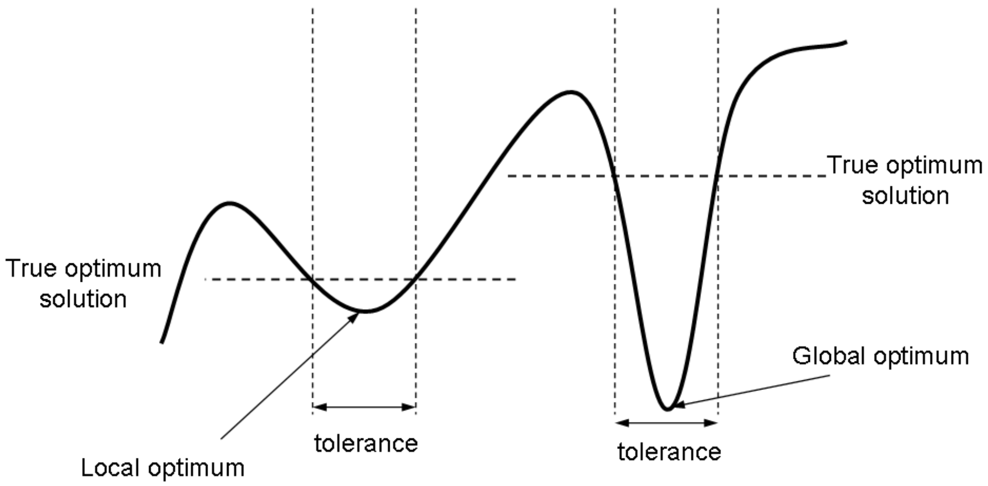

Figure 3: Change of the product dimensions within specified tolerances significantly worsens solution.

Analysis software in most cases is not created for the numerical optimization purposes. Although the computations are done on the float numbers, or even with double accuracy, the results are written to the output files with reduced accuracy, which is better than good for an engineer. Finite accuracy of the output often causes problems if finite difference method for derivatives computation is used. Very small difference in the models analyzed isn't reflected in the results, which from the computations point of view seem to stay the same Figure 5. As a result the derivative is equal to zero, and algorithm is very likely to stop the optimization.

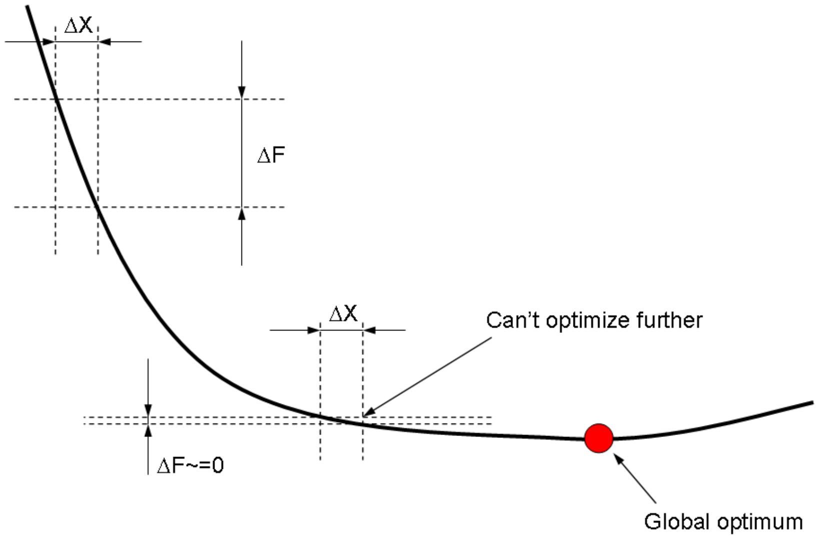

Figure 4: Too low output accuracy.

Numerical model for analyses also has finite accuracy, which can depend on analysis software's own convergence. Unfortunately this reduced accuracy of the returned results can cause similar problems as the finite accuracy of the numerical model. Obtaining the optimum point might require only subtle changes of the design parameters, but the analysis results, after the iterative convergence process, might have large scattering Figure 5. In that case the sensitivity of the analysis for the optimization algorithms is not good enough.

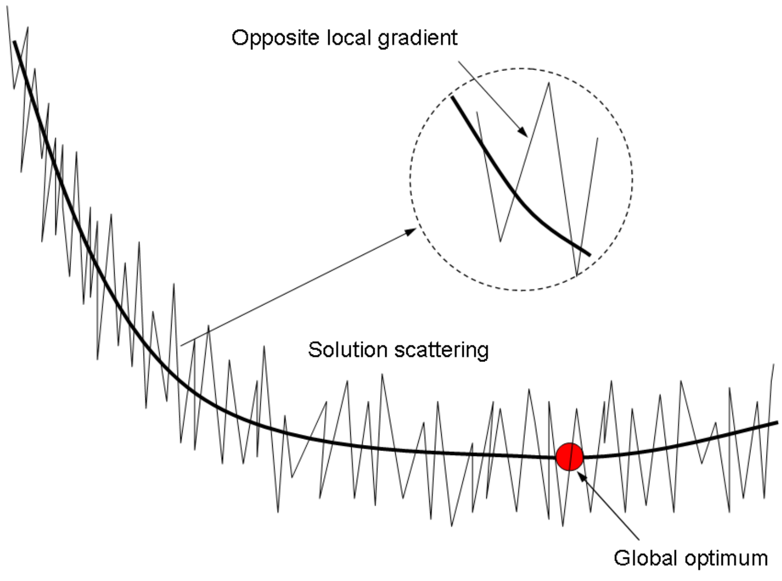

Figure 5: Scattering of the numerical solution is larger than sensitivity needed to obtain optimum.

Few of the optimization problems can be omitted, or reduced by usage of appropriate optimization algorithms. Gradient based methods can be very efficient compared to other optimization methods, but have also disadvantages. First well known is tendency to converge to the nearest local optimum. In practice, in many cases this is not a significant problem. Optimization tasks have often single best solution, and even if not, the designer can first choose good starting point for the algorithm, basing on experience, which is sufficiently close to the searched optimum.

Second global problem is obtaining well defined search direction, which will lead to the optimum solution. If the optimization algorithm is highly coupled with the analysis tools, gradient computations with adjoint methods [7] can be used. But very often professional software for analysis is commercial, without access to the source code. In that case the analysis tools have to be treated as a black boxes, with limited information about computational process. In that case optimization solution can be sensitive to the problems described earlier: too aggressive changes of the optimization variables, scattering results from multiple simulations, output values with insufficient accuracy. Solution might be heuristic algorithms like: Monte Carlo, Genetic Algorithms, Particle Swarm Optimization, and other. On the other hand heuristic algorithms are related to probability issues. There is no guarantee of finding the optimum, and the accuracy of the optimum is usually lower than with usage of the gradient algorithms.

2.2 Simple example of encountering optimization problems

This simple example, of finding airfoil's Selig S2027 maximum angle of attack, will show some of the outlined issues. Motivation for searching for the maximum angle of attack was possibility of the airfoil's geometry optimization. In that case maximum angle of attack would also change, and it is one of the airfoil's critical parameters. For the demonstration only the geometry of the airfoil is constant, and optimization algorithm is used to find the maximum point of airfoil's lift characteristics. Aerodynamic analysis of the airfoil are done with Xfoil software [8]. Figure 6. shows computed characteristic of the lift coefficient vs. angle of attack for the airfoil.

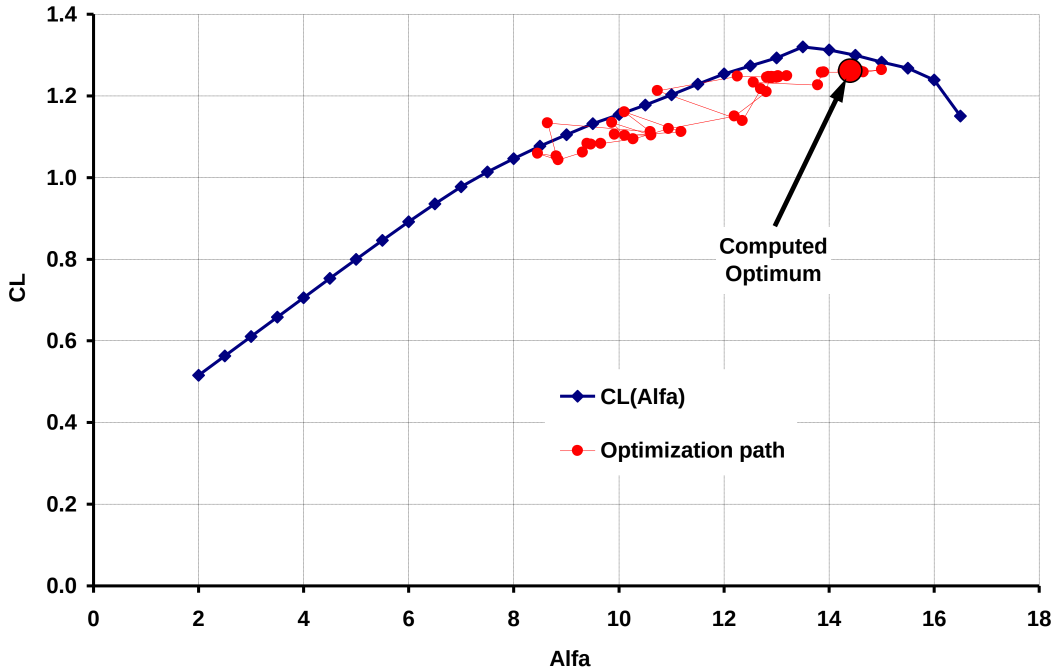

Figure 6: Lift coefficient characteristic vs. angle of attack.

Simple Newton-Rapson gradient algorithm was used to find the angle of attack for the maximum lift coefficient (1). The method finds zero value of the objective function. Knowing that the first derivative of the airfoil characteristic of lift coefficient from angle of attack will have zero value for the maximum, this could be the objective function (2). Than recursive equation for finding the maximum value of lift coefficient from angle of attack is defined as (3). For such stated problem the derivatives, compute with finite difference method, needed for equation (3) will be defined as (4) and (5). History of convergence of the problem is shown on Figure 7.

$$CL(\alpha)_{\max} \tag{1}$$

$$CL'(\alpha) = 0 \tag{2}$$

$$\alpha_{new} = \alpha - \frac{CL'(\alpha)}{CL''(\alpha)} \tag{3}$$

$$CL' = \frac{CL(\alpha + d\alpha) - CL(\alpha - d\alpha)}{2 \cdot d\alpha} \tag{4}$$

$$CL'' = \frac{CL(\alpha + d\alpha) - 2 \cdot CL(\alpha) + CL(\alpha - d\alpha)}{d\alpha \cdot d\alpha} \tag{5}$$

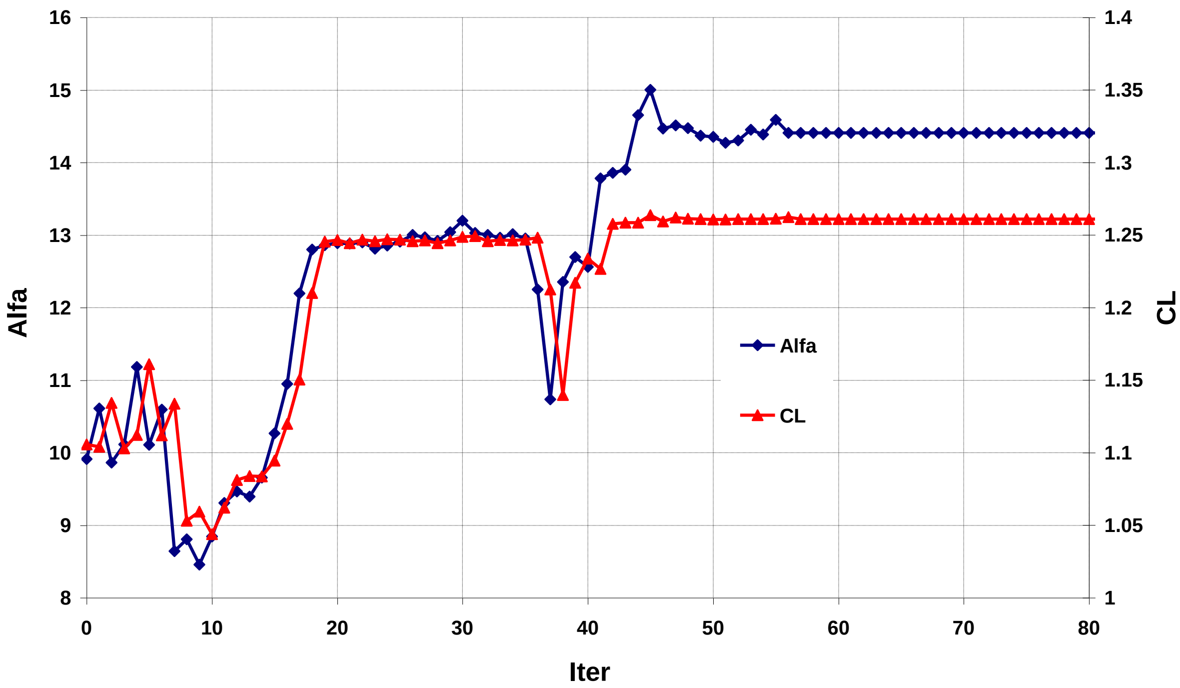

Figure 7: Optimization parameters history from Iterations.

Table. 1 Solutions comparison

	Alfa	**CL**
Optimization	14.41	1.2611
Direct analysis	13.5	1.3202

Although, the curve for which maximum point seems to be quite simple the history of optimization Figure 7 reveals some problems of the algorithm to reach the optimum. Output of the Xfoil is limited to four digits accuracy, and the optimization derivatives are computed by usage of the finite difference method. This limits the accuracy of the derivatives, and the problem pronounces strongly near the optimum. When the algorithm got on the right track from 10 to 20 iteration the improvement was clear, but while it got close the optimum the optimization solution began to scatter.

Looking on Figure 6 difference between points obtained from direct Xfoil analyses and during the optimization is noticeable. This is probably due to the different Xfoil convergence during the analyses. The problem was shown on Figure 5. Xfoil has strong coupling of the boundary layer solution with potential flow solution. It iterates to find improve coupling between the solutions. Moreover, it can use results obtained from previous analysis to estimate good starting point for the computations of the next angle of attack. During the optimization this history of computation of previous angle of attack isn't available and the convergence is worsened.

What is more relaxation factor had to be introduced in equation (3), because too aggressive behavior of the algorithm crushed the optimization process. All of that caused significant number of iterations to converge. Finding of the appropriate relaxation parameter also took some time.

Finally the difference between maximum lift coefficients from optimization, and from direct airfoil analysis, made by 0.5deg, differ less than 5%, which is quite good relation. But looking on the number of analyses executed, to find the optimum the case shows clearly superiority of the direct analysis in that case. The direct analysis could be done even denser, by for example 0.1deg to get better estimation of maximum lift coefficient, and even then the computational cost of optimization won't be reached.

If estimation of the maximum lift coefficient is only part of a much bigger optimization problem, in which conceptual methods of analysis are used, there is no point to obtain the maximum lift coefficient of an airfoil with great accuracy. The designer might assume that initial estimation of the maximum lift coefficient is good enough, and it might be kept constant without big influence on the global solution. This way optimization method of finding maximum lift coefficient, or subsequent airfoil analysis is only needed prior to the global optimization computations.

3 Executing external program for analysis

There are number of ways to execute external simulation analysis, but all programs should satisfy few conditions:

- software for analysis has to be able to work in batch mode
- ability to pass data to the simulation software (through input file in most cases)
- result of simulations have to be returned (through output file in most cases)
- optimization software has to wait for simulation software till the analysis finish

To execute external program with parameters different system calls can be used, that slightly differ in the behavior. Most programming languages are able to execute system shell procedure, for example in C++ this is done, by: system("system command"). In windows environment, system commands that execute program are:

- **cmd** - opens new instance of system commands interpreter, as a new process, optimization software won't wait for the shell to end
- **start** - starts separate command window interpreter and in new process, and executes specified program, optimization software won't wait for the shell to end
- **call** - starts specified program in the same process, control of computations is passed to the newly opened program until the analysis software finishes computations and closes, than the control is passed back to the parent program, if simulation computations crush the optimization might also crush

The other way to execute external software is to use system API procedures, that can be used directly in the optimization software.

- **CreateProcess() -** starts specified program as a new process, among input parameters user can define if the optimization software should wait till the process ends

3.1 Multithreading

Optimization process always involves multiple objective function executions. Parallel computations of objective function can improve significantly time of computations, but also introduce new issues, that have to be solved.

3.1.1 Cross platform libraries

Parallel computations need appropriate libraries that will manage multithreading. That could be done by dedicated system functions, but then the optimization is restricted only to particular system. There are few libraries that can handle multithreading on different systems, for example: pthread library (posix threads library), which is capable to work on the most popular systems like Windows, MacOS, and Linux.

3.1.2 Input and output files management

Another challenge is files management. Assuming that for the analysis external programs are used, the programs will need unique files for every thread input, and they also have to produce unique output files. Otherwise it is possible that at the same time different threads will read, and write to the same files, what will lead to optimization algorithm unpredictable behavior.

3.1.3 Passing information in nested to optimization

There are many multidisciplinary optimization architectures. One claimed to be the most efficient, is nested optimization architecture [9]. In this kind of optimization there is global optimization algorithm, and inside this algorithm another optimization process is running. For example aerodynamic optimization of wing can be driven by global optimization algorithm. But the aerodynamic loads will affect optimal solution of structure, and it's mass. In that case, after obtaining aerodynamic loads, strength of the wing can be optimized inside the global aerodynamic optimization process, and way mass of the wing can be derived. Global objective function containing aerodynamic efficiency and mass, which both influence range, can be build. Information about geometry, loads and other parameters, for every thread, have to be passed to the nested strength optimization process. This can be done by passing needed information while starting nested optimization process through the batch file.

4 Optimization management

Optimization management, which includes specifically: choosing right optimization algorithm for the optimization, definition of the optimization task, choice of the multidisciplinary optimization architecture, and parallel simulations is difficult if appropriate numerical tools aren't available. It was also motivation of the author of the article to create optimization software OptiM [10], which beyond pure optimization would also support user with common issues related to setting up the optimization process. Experience related to multidisciplinary optimization was gained while developing and using the optimization software Figure 8.

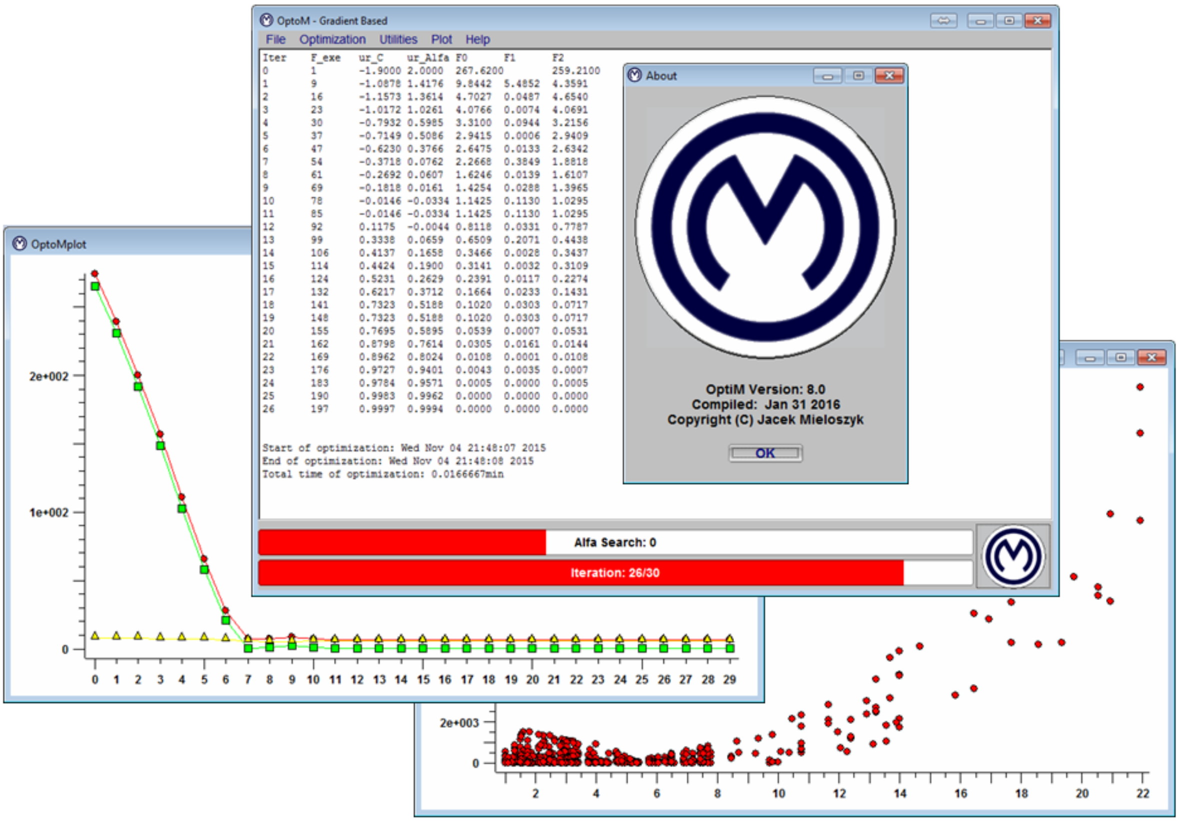

Figure 8: OptiM software layout.

4.1 Quick change of algorithms and algorithms development

Regrettably, there is no unique optimization algorithm that is appropriate for every optimization task. Problems may occur for example with directional gradient computation, or finding the solution too far from the optimum for heuristic optimization algorithms. It is beneficial to have possibility to quickly change the optimization algorithm. It might not be a common feature in the optimization software, because data management during automatic optimization process is also challenging.

4.2 Optimization task definition

User of the optimization algorithm doesn't have to be the person that created the optimization algorithm. An engineer will be focused mainly on solving and optimizing engineering problem. It is inevitable to know the basics of the optimization algorithms operation, but not how to write them from a scratch. The optimization software used [10] contains build in optimization algorithms and user only defines optimization task in a dynamically linked to the software library. User only has to modify files of the dynamic library and compile it. The modification might involve single line of code (e.g. mathematical expression), but it can also be very complicated optimization task, which involves external program calls for automated parallel analyses.

4.3 Multidisciplinary optimization strategy

There are multiple multidisciplinary optimization strategies that were described in a number of papers [11][12]. Here will be shown results of multidisciplinary optimization on an example of boxwing aircraft configuration Figure 9, which were done in the MOSUPS project [13][14].

Figure 9: Aircraft in boxwing configuration built in the MOSUPS project.

Figure 10 shows the diagram of the multidisciplinary optimization with nested strength optimization, which was run parallel. Optimization algorithm used was Particle Swarm Optimization (PSO). The PSO algorithm tries to modify swarm member's optimization parameters to improve their efficiency, in this case minimum power for flight. Aerodynamic and strength analysis were coupled in the optimization process. In every thread single configuration of aircraft was aerodynamically analyzed. The aerodynamic loads could be derived for the strength analysis. In the next step, inside nested optimization process, thickness of wing's surfaces was sized to get minimum mass of the structure for the particular aircraft configuration in the swarm. Obtained results from aerodynamic analyses and mass from strength optimization were used to compute global objective function value.

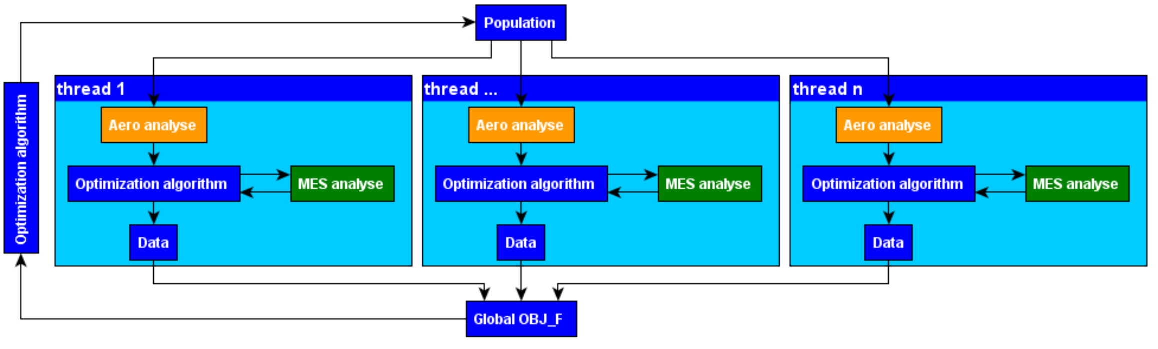

Figure 10: Optimization architecture, with nested optimization.

Figure 11 shows the first obtained results. The optimization solution is scattering. The remedy for this problem was increasement of the number of iterations in the nested optimization. This improved the accuracy of the obtained converged solution of the nested optimization, and as a consequence improved global optimization results. Figure 12 shows the improved results. Although, problem of convergence was solved, the time of optimization significantly increased.

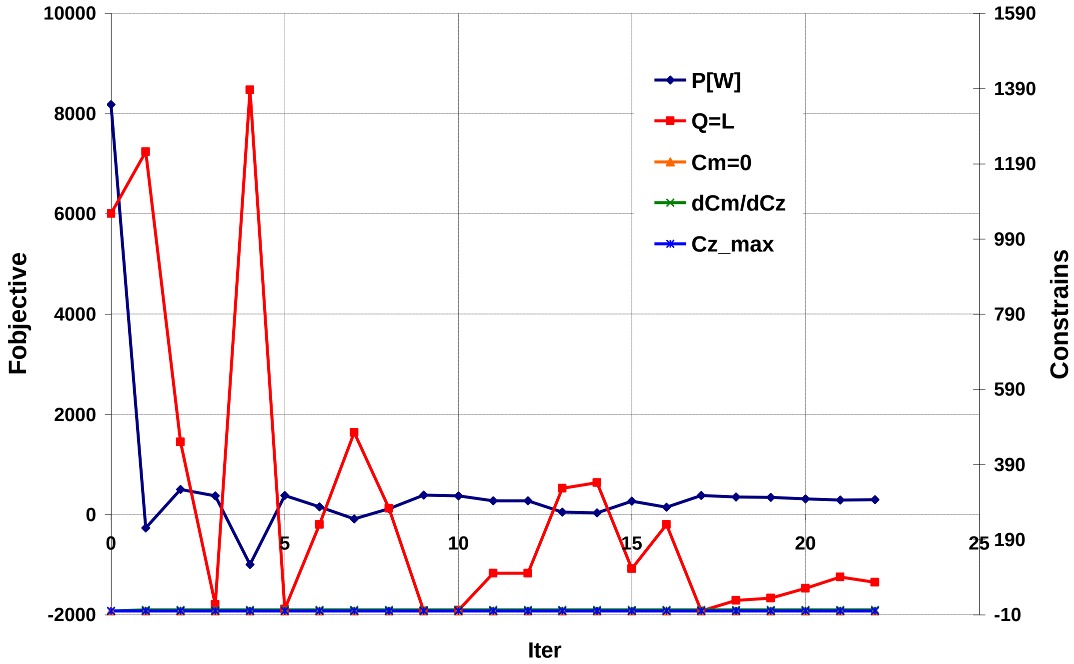

Figure 11: Initial results of the multidisciplinary optimization.

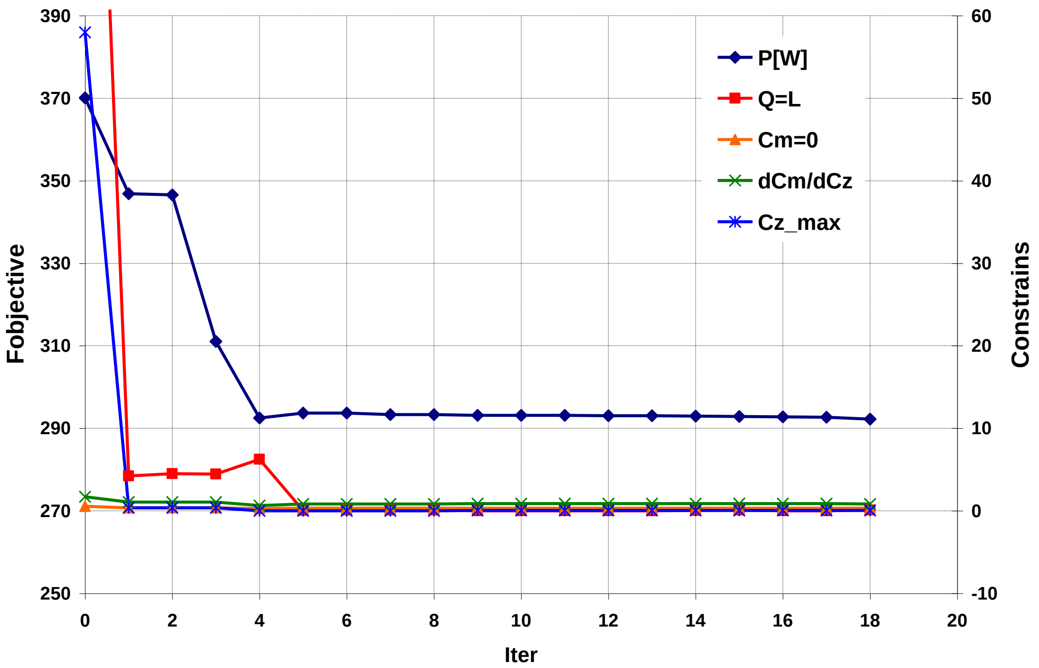

Figure 12: Final results of the multidisciplinary optimization after optimization parameters adjustment.

5 Conclusions

Lessons learned in application of multidisciplinary optimization were discussed in the article. Multiple issues related to applied numerical optimization were shown, in particular: computations efficiency, quality of the obtained optimum, external programs execution, parallel simulations for optimization, multidisciplinary optimization parameters setting.

Two examples show how the potential pitfalls that can reveal. First one is fairly simple, but shows most of the issues related to numerical optimization. Second example, applicable to solving real engineering problem shows the consequences of choosing one of possible multidisciplinary architectures.

Person responsible for optimization has to be very aware of the optimization techniques. Most of the encountered problems are solvable, but the applied solutions often have consequences, for example higher computational cost.

6 References

[1] Goetzendorf-Grabowski T., Mieszalski D., Marcinkiewicz E.: Stability analysis using SDSA tool, Progress in Aerospace Sciences, Volume 47, Issue 8, November 2011, Pages 636–646

[2] Mieloszyk J., Kalinowski M. “FEM analysis of joinwing aircraft cnfiguration”, proceedings of the 5th CEAS Air & Space Conference, Delft, 2015

[3] Mieloszyk J. “HANDLING OPTIMIZATION PROBLEMS ON AN EXAMPLE OF MICRO UAV”, Proceedings of the 3rd CEAS Air & Space Conference, 21st AIDAA Congress, Venice, 2011

[4] Goetzendorf-Grabowski T. Mieloszyk J., Mieszalski D.“ MADO – Software Package for High Order Multidisciplinary Aircraft Design and Optimization”, Proceedings of 28th International Congress of the Aeronautical Sciences, Melbourne, 2012

[5] Thomas H., Zhou M., Schramm U. “Issues of commercial optimization software development”, Structure Multidisciplinary Optimization, 23, 2002, pp. 97-110

[6] Ahn J., Kwon J. H. “An efficient strategy for reliability based multidisciplinary design optimization using BLISS”, Structural and Multidisciplinary Optimization, May 2006, Volume 31, Issue 5, pp 363–372

[7] Leoviriyakit K. “Wing Planform Optimization via an Adjoint Method”, Stanford University – doctoral thesis, 2005

[8] Drela M., (1989) “XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils”, in Low Reynolds Number Aerodynamics, Ed. T. J. Mueller, Lecture Notes in Eng. 54

[9] Asymmetric

[10] Mieloszyk J., OptiM - Software package for numerical optimization, Warsaw University of Technology, http://www.meil.pw.edu.pl/add/ADD/Teaching/Software/OptiM P.

[11] Joaquim R. R. A. Martins, Andrew B. Lambe. "Multidisciplinary Design Optimization: A Survey of Architectures", AIAA Journal, Vol. 51, No. 9 (2013), pp. 2049-2075.

[12] Nathan P. Tedford · Joaquim R.R.A. Martins, “Benchmarking multidisciplinary design optimization algorithms”, Optimization and Engineering, February 2010, Volume 11, Issue 1, pp 159–183

[13] Mamla, C. Galiński “Basic induced drag study of the joined-wing aircraft”, Journal of Aircraft, Vol. 46, No 4, July-August 2009, pp. 1438-1440

[14] Galinski C., Hajduk J., Kalinowski M., Wichulski M., Stefanek Ł., „Inverted Joined Wing Scaled Demonstrator Programme”, Proceedings of the ICAS 2014 conference, St. Petersburg, 7-12 September 2014