Adjoint Optimization combined with mesh morphing for CFD applications

Alberto Clarich*, Luca Battaglia*, Enrico Nobile**, Marco Evangelos Biancolini°, Ubaldo Cella°°

*ESTECO Spa, Italy. Email: engineering@esteco.com

**University of Trieste, Italy. Email: nobile@units.it

°University of Rome "Tor Vergata", Italy. Email: biancolini@ing.uniroma2.it

°°Design Methods, Italy. Email: ubaldo.cella@designmethods.aero

Summary

This paper aims to investigate the applicability of Adjoint optimization combined with mesh morphing to the industrial practice, by the integration of commonly used industrial-grade simulation software.

Adjoint techniques are efficient optimization methods in terms of accuracy of results and short computational cost, but normally are limited to in-house simulation codes, that allow the calculation of partial derivatives of the observable quantities within the model simulation. Quite recently, some CAE packages introduced this capability within their solvers. This is the case of ANSYS Fluent, which makes available the derivatives of a given observable (objective) as a function of mesh points coordinates. This, combined with a mesh morphing tool such as RBF Morph, allows to automatically compute the derivatives in function of design parameters, such as the amplifications of RBF solutions that control the shape of the mesh.

Integrating these software in the optimization platform modeFRONTIER from ESTECO, it becomes possible to apply efficient gradient based optimization algorithms, with the big advantage of a full automatic process integration, the possibility to exploit the post-processing tools available in modeFRONTIER, and a very short number of design simulations to optimize the objective function. Methodology details and CFD application benchmarks will be illustrated.

Keywords

Optimization, Adjoint, CFD, Mesh morphing

Adjoint optimization combined with mesh morphing

Optimization based on Computer-Aided Engineering (CAE) simulations plays a central role in many engineering and scientific fields. In the modern competitive scenario, being able for companies to keep up the pace and to offer innovative design solutions with advanced, efficient and cutting edge technologies, is of primary importance [1].

In this context, gradient-based methods [2] have emerged as a powerful tool able to drive design, exhibiting good performances and speed also in the optimization scenarios characterized by a high

number of design variables. The main limitation of these methods relies on the necessity to evaluate partial derivatives of the objective function with respect to design parameters; if sensitivities are not available from the simulation software, it is necessary to approximate them by finite difference evaluations, increasing significantly the computational task as the number of parameters grows.

Conversely, some commercial software such as ANSYS Fluent are giving the possibility, through the Adjoint evaluation, to economically obtain sensitivities values of the defined observable quantity, in function of mesh points coordinates. Another software, such as RBF Morph [3], commonly used to modify the mesh shape by the usage of morphing parameters [4], when combined with Fluent allows the possibility [5] of transforming, by chain rule, the observable sensitivities as function of mesh coordinates ($x_{j,node\_i}$ in eq.1 below) to sensitivities as function of morphing parameters, i.e. the design variables of the optimization problem ($x_{j,param\_k}$ in eq.(1) below).

$$\delta F = \sum_{i=1}^{nodes} \sum_{j=1}^{3} \frac{\partial F}{\partial x_{j,node\_i}} \delta x_{j,node_i} = \sum_{k=1}^{parameters} \sum_{j=1}^{3} \frac{\partial F}{\partial x_{j,node\_i}} \frac{\partial x_{j,node\_i}}{\partial x_{j,param\_k}} \delta x_{j,param\_k} \quad (1)$$

This allows to set up an optimization workflow, using a software such as modeFRONTIER [6], where any gradient-based optimization algorithm can read directly the sensitivities in function of the design variables (morphing parameters), without any additional simulation required. For each iteration of the algorithm all the derivatives, independently from the number of shape parameters, are computed at the cost of a single adjoint simulation, resulting in a very low global number of design simulations to reach the optimum solutions of the problem.

Another advantage of this methodology, from the point of view of its applicability to industrial practice, is the possibility to optimize any simulation model without the need of modify and parameterize the source CAD. In many industries, CAD departments are in fact independent from the simulation departments, therefore if the latter ones need to optimize the shape of any component, this can be possible just by applying a mesh morphing tool on the baseline model. The shape obtained at the end of the process can be converted into a draft CAD version exploiting RBF Morph back2cad tools and used for the generation of the final geometrical model.

In this paper we will first illustrate the methodology to be followed to easily set up an adjoint optimization problem based on morphing parameterization using the commercial software mentioned above. The efficiency of the methodology will be evaluated as a function of a different number of parameters, using a simple CFD test case. At the end, the methodology will be applied to a realistic industrial benchmark, consisting in the shape optimization of a heat exchanger manifold, with the objective of guaranteeing a uniform mass flow across the channels.

Setup of adjoint solver with morphing parameters

The first application case we have implemented to describe and test the methodology is the optimization of the shape a U-bend diffuser pipe, with the target of minimizing pressure drop.

The model is defined in ANSYS Fluent (fig.1): the flow is laminar (Re_D=500), and a static pressure condition is applied to the outlet surface (0 Pa). The mesh is controlled by the RBF Morph Add On, that allows to modify the mesh according to radial basis functions (RBF) mesh morphing. The distribution of RBF points is defined on the surface of cylinders that act as sculpting tools. For this application, we have defined scaling rules (in both cross directions) for the points on the cylinders along three sections, fixed positions for the points on cylinders at the beginning and at the end of the

curve and finally a sixth cylinder to gain the an overall change of the whole curve for a total of 7 shape parameters, which control the shape of the cross section of the pipe along the bend [7].

In order to set up the adjoint computations and transform the derivatives of the observable function (here the pressure drop) in function of the morphing parameters, it is just enough to define few commands, that will then be used by modeFRONTIER as a template to repeat the process automatically for any configuration proposed by the optimization. The main commands can be identified as it follows:

- (define *namevarx valuevarx*): for each variable x, define its name and the baseline value
- (define *numiter* 400); (define *numiteradj* 150): define max number of CFD and adjoint iterations
- (Ti-menu-load-string (format #f "*name.cas*")): load the Fluent case file
- (my-open-udf-library rbf-library): load RBF Morph libraries
- (Ti-menu-load-string (format #f "it ~a" *numiter*)): run CFD computations
- (Ti-menu-load-string (format #f "/adjoint/run/iterate *numiteradj*")): run adjoint
- (Ti-menu-load-string (format #f "/adjoint/reporting/report inner")); (rbf-smorph-init): initialize
- (rbf-smorph-adjoint '((*namevarx* 1))); (define *adj-varx* (%rpgetvar 'rbf/smorph-adjoint-eval)): apply morphing for any variable x, and create a sensitivity variable
- (define gradfile (open-output-file "*Gradients.out*")); (display "Sensivitity to varx " gradfile) (write *adj-varx* gradfile) (newline gradfile); (close-output-port gradfile): open an output file and save each sensitivity of any variable x
- (Ti-menu-load-string (format #f "/define/parameters/output-parameters/write-to-file pressure-drop-op *pressure-drop.out*")): save observable to another file

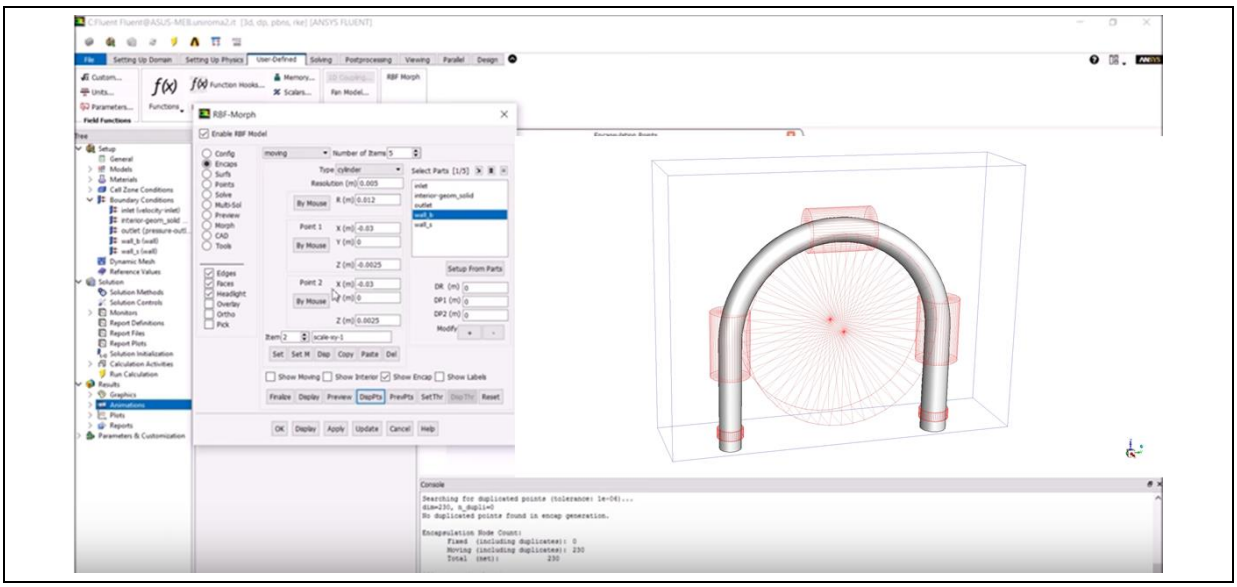

Figure 1: Morphing of U-bend diffuser pipe

At this point the model file is ready to be implemented in modeFRONTIER workflow for the automation process definition.

Optimization workflow setup: modeFRONTIER

Figure 2 reports the workflow that has been implemented in modeFRONTIER to automate the process execution and define the optimization loop. In the first row each node represents one of the 7 input parameters which control the morphing parameters: their range of variation has been defined in order to amplify the morphing from the minimum to the maximum value allowed. These have been selected in order to guarantee a good quality of the mesh during the morphing step. The input nodes are linked directly to the application node, the *Easydriver* node, which uses a template file for the automation of Fluent and adjoint computations. Each variable is in fact linked to the position of the script file, defined in previous section, where its value is assigned. The driver panel of the node defines the command to execute the simulations in batch mode (in DOS the command is simply "*fluent.exe –i script.scm"*, where *script.scm* is the name of the script file described in sec. 1). Finally another template is used to extract the output parameters (observable value and its derivatives) from the output files defined in the script, specifying also in this case their relative positions.

At the end, in the bottom part of the workflow, each output parameter node extracted from the application file is linked to the gradient objective node, in which it is specified what is the objective function (the observable pressure drop) and which is its derivative for each input variable.

The optimization can at this point be run choosing one of the available gradient based algorithms available in modeFRONTIER (among which, B-BFGS [8], Levenberg-Marquardt, and SQP proprietary implementations for single and multi-objective problems, continuous and mixed variables). the first application case we have implemented to test the methodology is the shape optimization of the U-bend diffuser pipe described in the previous section, with the target of minimizing pressure drop.

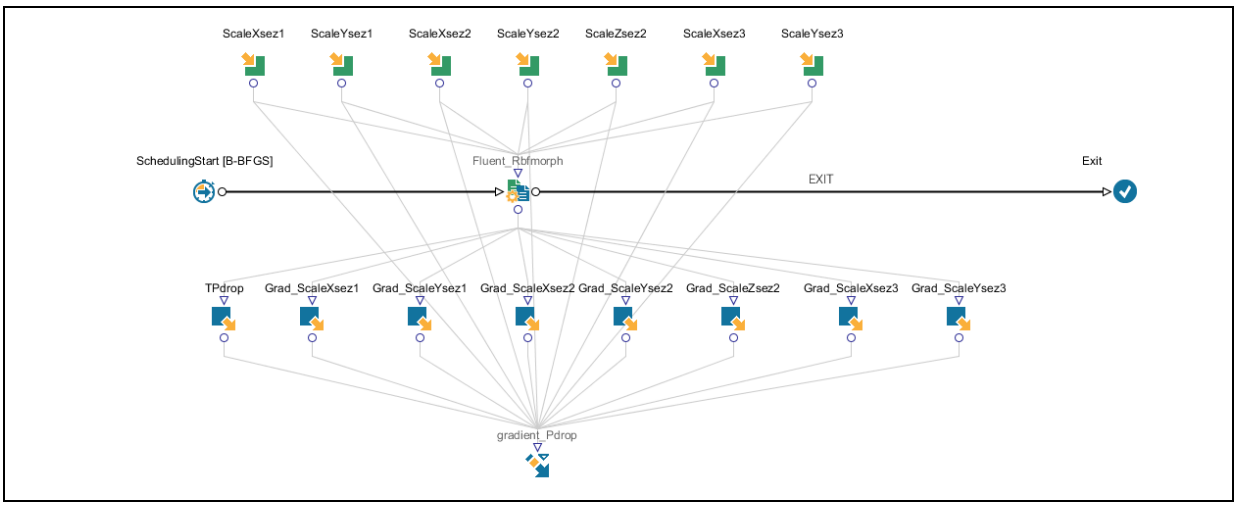

Figure 2: modeFRONTIER workflow to automate the optimization process

Adjoint advantages versus parameters number

In order to prove the efficiency of the proposed methodology, the same optimization problem has been analyzed using both a classical approach (Genetic Algorithm or GA) without the usage of the gradient, and the gradient-based methodology described in previous sections.

To compare the efficiency of the two approaches, we have defined a total number of simulations for each approach in such a way that the overall computation time could be approximately the same.

Since one CFD evaluation (400 Fluent external iterations) takes about 4 minutes while the simulation completed by adjoint (400+150 additional iterations) takes about 13 minutes, we have to consider that for the adjoint approach each design takes a simulation time about three times higher than that for the classic one. Therefore, if we have defined 300 as overall number of simulations for the GA approach (since by the practical experience [9] 20 generations of 15 designs each one is considered efficient to solve a single-objective problem of 7 design variables), we need to define a total number of simulations for the gradient-based approach not much higher than 100.

Despite the fact that gradient-based approaches are very fast, their main limit is the fact that they can converge to a local minimum, depending on the choice of the starting point. For this reason, in order to improve the robustness of the methodology, we have decided to make the algorithm start from the same points of the starting DOE (Design of Experiments) defined for the GA approach, therefore leaving a maximum of 7 iterations for each of the 15 DOE points (with 100 simulations, the overall computational time will be the same for the two approaches).

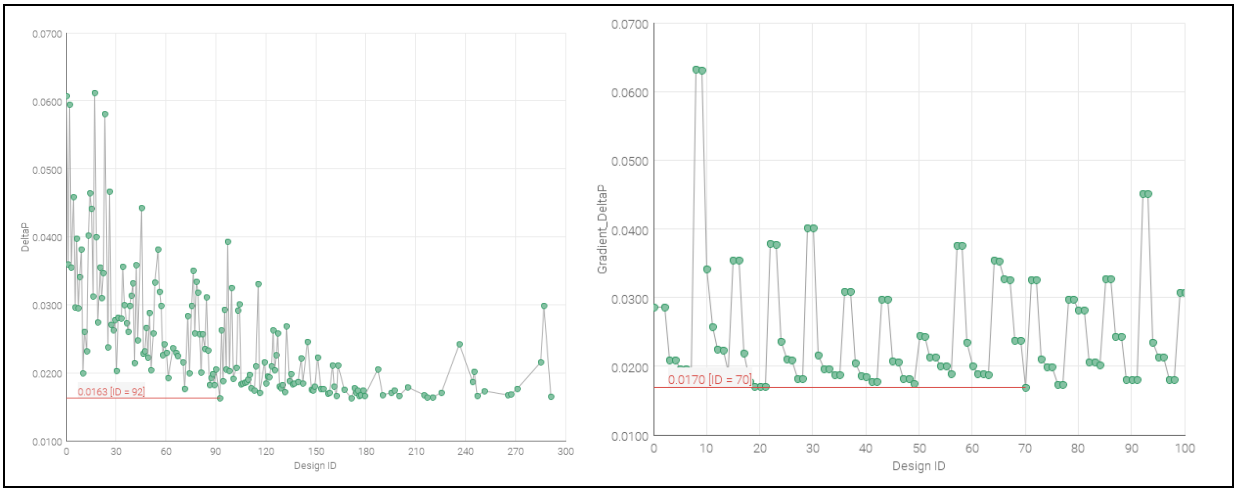

Figure 3: Optimization convergence: classical GA (left) and gradient-based approach (right)

The results of the two methodologies, in particular the convergence history of the pressure drop objective function, are defined in figure 3. We can note that the GA (fig.3 left) takes just about 90 simulations (therefore 360 minutes) to reach the global minimum, while the gradient approach (fig.3, right) takes just 7 simulations (therefore 90 minutes) practically whatever is the starting point of the DOE (the points with higher value of objective functions in the chart) used to initialize the gradient.

The gradient-based method is therefore about 4 times faster than the classical GA to reach the global optimum. This advantage is particularly evident when the number of parameters increases, since if the simulation time does not change much for the gradient approach, it increases linearly for algorithms based on evolutionary process.

To prove this advantage, we have repeated the analysis for this model changing the number of parameters, fixing as constant some of them to reduce the number, or adding other morphing parameters (i.e. number of controlled sections) to raise it. The expected behavior is confirmed by the results obtained (fig.4). The two approaches are compared, in terms of overall simulation time required to reach to global optimum, and the advantage of the gradient approach increases with the number of parameters. The number of iterations for the adjoint approach here increases just because of the higher influence of local minima to the algorithm convergence: to reach the global optimum, it is necessary to explore a DOE of an higher number of points (for the case with 12 parameters in fig.4, we have defined a DOE of 15 points, then applied 7 B-BFGS iterations starting from the best point of

the DOE to reach the same global optimum that has been reached by the GA in 168 iterations: the CPU time ratio is 672 minutes for the GA, and 264 minutes for the 22 simulations completed by the gradient approach).

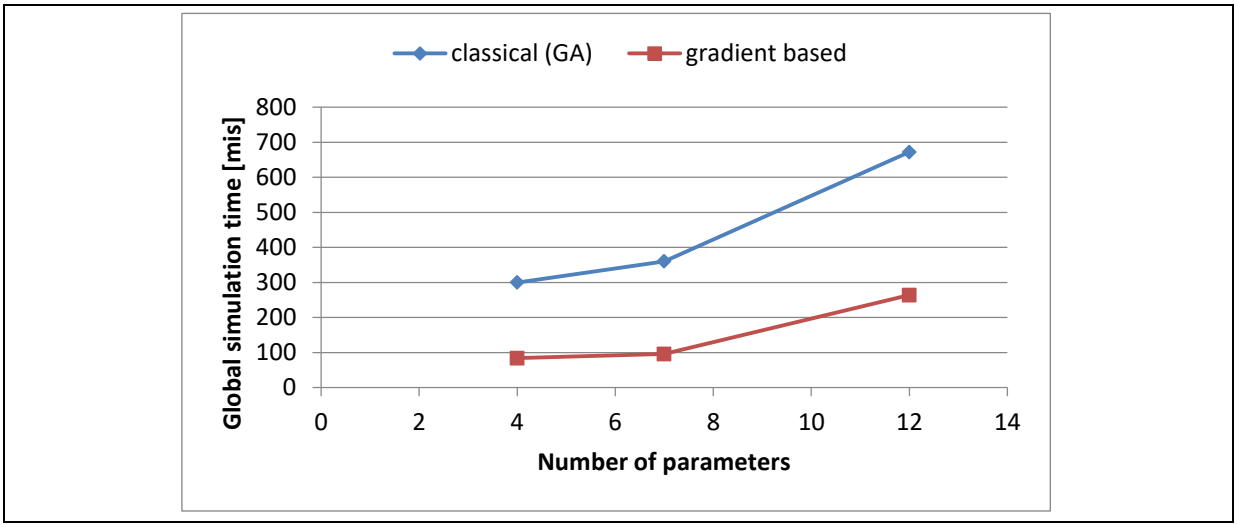

Figure 4: Influence of parameters number on the convergence speed (CPU time in minutes)

Heat exchanger manifold optimization

The important conclusions we have obtained in the previous section could have been partially influenced by the relative simplicity of the model considered, e.g. steady laminar flow, and the linearity of the objective function. For this reason, we have applied this approach to a different application, the optimization of the shape of an intake manifold of a heat exchanger (fig.5: Re_D=50.000, $k-\varepsilon$ turbulent model).

The baseline shape is not optimal, since even though the mass flow is split in a pretty uniform way to the 10 outlet channels (as we can note in fig.5), the flow is not fully developed as it enters in each channel (see for instance channels from 2 to 6 from the left): a longer development length causes an increase of pressure drops in the channels and a worse heat exchange performance [10].

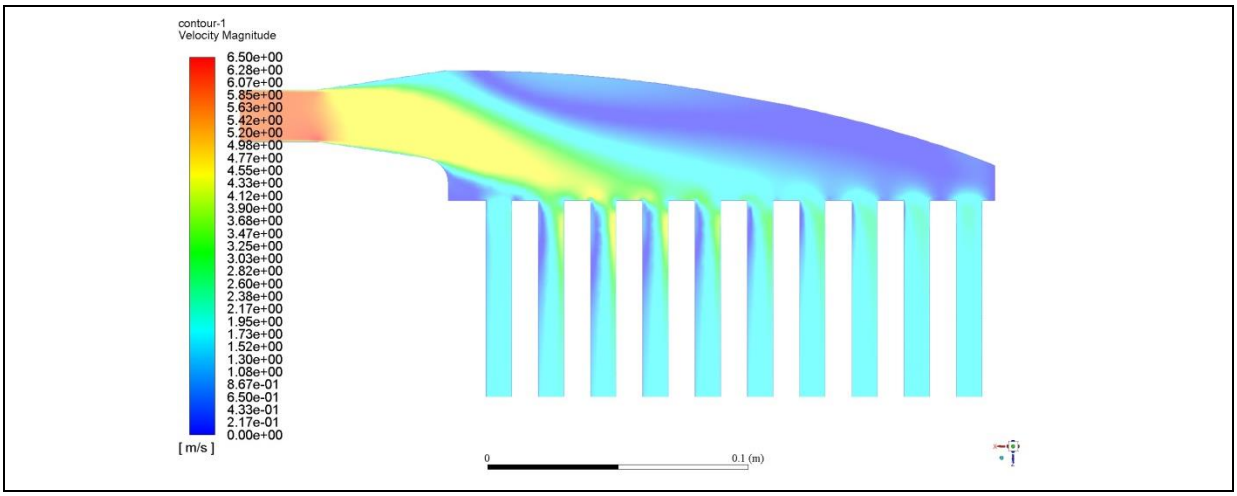

Figure 5: Velocity field in heat exchanger manifold

We have therefore computed an adjoint simulation, defining as observable to be optimized the sum of the standard deviation of velocity field at the inlet of each outlet channel, in order to optimize the uniformity of the velocity field of each channel at its inlet; at the same time, to keep an uniform split of mass flow as good as the baseline, we have added a constraint for each channel imposing that mass flow deviation from the average one is less than the one of the baseline (0.03 kg/s). The objective and constraints are therefore defined as:

$$\min \sum_{i=1}^{10} \int_{sez\_i} (v - \bar{v})^2 dS \qquad \left| \int_{sez\_i} \rho v dS - \bar{\Phi} \right| < 0.03 \qquad i = 1,..10 \tag{2}$$

At this point, to define an efficient set of morphing parameters, we have observed the sensitivity field (function of the mesh points) as obtained by the adjoint simulation of the baseline. As we can note from fig.6, the regions where the sensitivities are higher correspond to the area where the inlet pipe is merged to the manifold, and to some channels point of contact with the manifold.

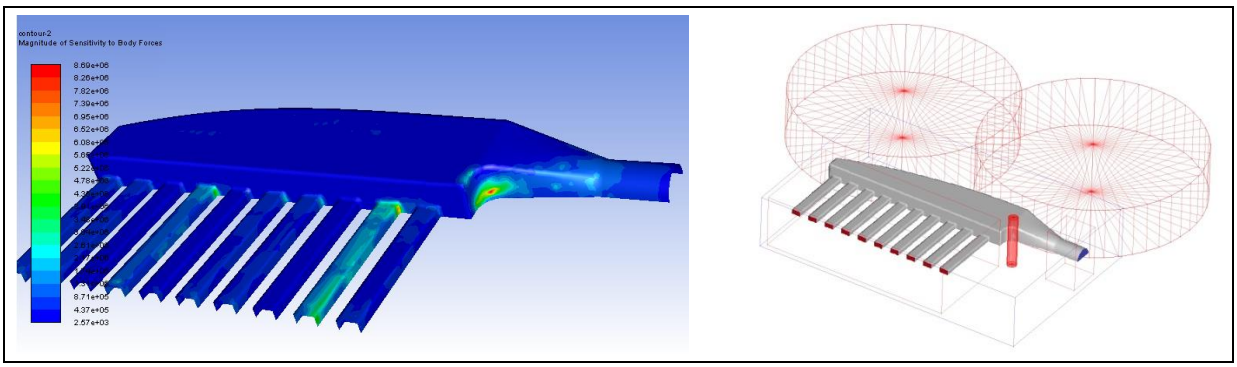

Figure 6: Sensitivity field of baseline design (left) and definition of morphing parameters (right)

These ones are the areas where it is expected to find the highest variation of the objective function, for this reason we have defined (fig.6, right) across these region some cylinders (left and right parts in the top curve of the manifold, and bottom part of the inlet pipe) and other morphing surfaces (in correspondence of the interface sections between outlet channels and manifold), to modify the shape of the mesh in these areas. The morphing entities are represented in fig.6, for a total of 23 control parameters, defined by the scale factors of the cylinders radius and the scaling in two directions of the channel surfaces.

The gradient-based approach described in previous sections has therefore been applied to this model. For the optimization process, it has been applied a B-BFGS algorithm, that has reached the convergence after a total number of about 50 iterations. To speed up the convergence, the process has been split into two phases.

In the first phase, only the three variables related to the manifold shape have been considered, obtaining a convergence after 15 iterations with a reduction of the velocity field variance at the channels inlet from 12.08 m^2/s^2 to 9.57 m^2/s^2 (-20%). The best configuration found after this first phase has been used as starting point for the following phase, in which the adjoint simulation has been set up in function of the 20 parameters controlling the channels inlets. After 30 further simulations, the velocity field variance has been further reduced to 7.72 m^2/s^2, still respecting the constraint on the mass flow split, for a global reduction of the objective function with respect to the baseline configuration of over 36%. Figure 7 reports the velocity field in the optimized configuration, which can be compared with the baseline represented in fig.5.

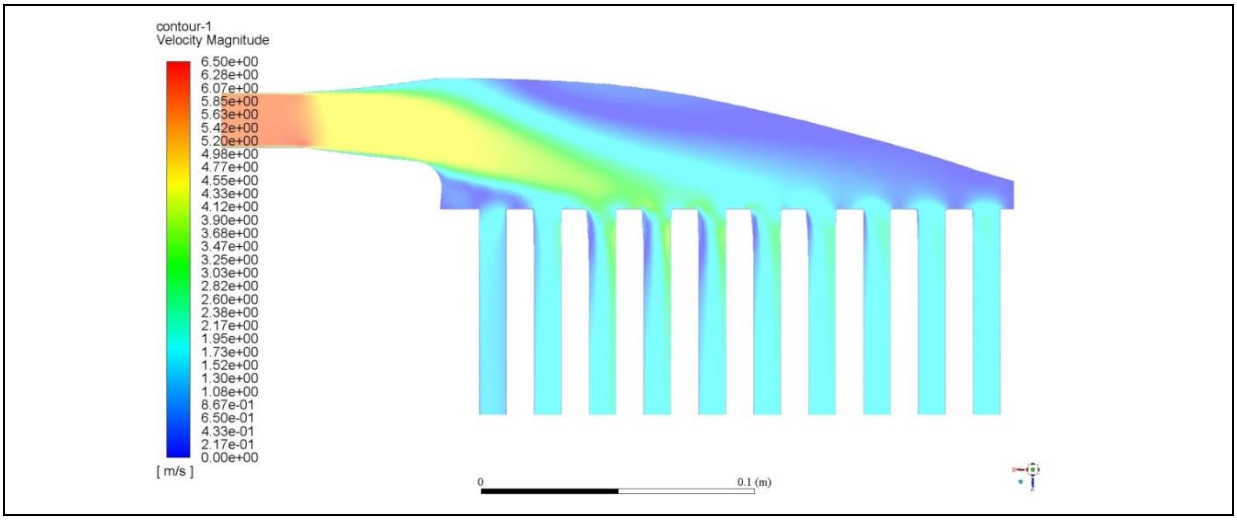

Figure 7: Optimized shape of manifold; compare velocity field with baseline (fig.5)

Conclusions

In this paper we have described, tested and applied to a CFD multi-objective optimization problem a methodology based on adjoint simulation combined with morphing parameterization and gradient-based optimization algorithm. The advantage of the approach, which has proved to be highly competitive in terms of simulation time and efficiency of results compared to other classic approaches, relies on the fact that it can be applied linking together CAE software of large industrial diffusion, such as ANSYS, RBF Morph and modeFRONTIER, and it is not an intrusive method requiring to modify simulation codes, but is very simple to implement for any industrial application just using the available GUI interfaces.

References

[1] Tormanen M., “Effective Optimization in Early Development Phases”, Procs. of ESTECO International Users Meeting, May 2018
[2] Yuan, Ya-xiang, “Step-sizes for the gradient method”, AMS/IP Studies in Advanced Mathematics, Providence, RI: American Mathematical Society. 42 (2): 785, 1999
[3] http://www.rbf-morph.com/
[4] Biancolini ME, How to boost fluent adjoint using RBF morph. In: Automotive simulation 781 world congress 2014 on 9–10 Oct 2014 in Tokyo, Japan, 2014
[5] Biancolini M.E., “Fast radial basis functions for engineering applications”, Springer, 2018. Chapter 8 “Adjoint Sensitivities and RBF Mesh Morphing” pp. 137-174.
[6] http://www.esteco.com/
[7] https://www.youtube.com/watch?v=3DNDXAG56os&feature=youtu.be
[8] Fletcher R., “Practical methods of optimization (2nd ed.)”, New York: John Wiley & Sons, 1987
[9] Poloni C., Pediroda V., “GA coupled with computationally expensive simulations: tools to improve efficiency”, 1997
[10] R. Russo, E. Nobile, P. Ranut, “modeFRONTIER for Virtual Design and Optimization of Compact Heat Exchangers”, Conference Paper in SAE Technical Papers, October 2014