ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

SHAPE OPTIMIZATION OF TURBOMACHINERY ROWS USING A PARAMETRIC BLADE MODELLER AND THE CONTINUOUS ADJOINT METHOD RUNNING ON GPUS

K. T. Tsiakas[1], **F. Gagliardi**[1], **X. S. Trompoukis**[1], **and K. C. Giannakoglou**[1]

[1]National Technical University of Athens, School of Mech. Eng., Lab. of Thermal Turbomachines,
Parallel CFD & Optimization Unit, Athens, Greece
e-mail: {tsiakost,fl.gagliardi,xeftro}@gmail.com ,kgianna@central.ntua.gr

Keywords: Turbomachines, Blade–Row Parameterization, Adjoint Optimization, GPUs.

Abstract. *This paper presents a gradient-based shape optimization method for turbomachinery rows. The developed continuous adjoint method for incompressible flows, with a fully differentiated turbulence model, is coupled with an in-house 3D blade parameterization software, which is differentiated to support the gradient-based optimization process. The in-house flow and adjoint solvers are implemented on a cluster of NVDIA Graphics Processing Units (GPUs). The parameterization software creates the blade by superimposing thickness on both sides of the mean-camber surface. The design variables are NURBS coefficients which ensure smooth shape changes during the optimization, despite the great number of degrees of freedom. In addition, NURBS surfaces are used to describe the final shape. Geometric sensitivities, which stand for the ratio of boundary displacements over the corresponding variation in any of the CAD parameters, are computed by differentiating the parameterization software. Based on the chain rule these are combined with the gradient of the objective function with respect to (w.r.t.) the displacements of the blade or casing nodes, as computed by the adjoint method, and used to update the design variables. During the optimization, the grid is deformed according to the updated shape of the flow domain; regenerating the grid is avoided, by making use of the NURBS surface point inversion technique to retrieve the new surface grid and, then, Radial Basis Functions (RBFs) to propagate the displacement of the boundary nodes to the interior of the computational grid.*

1 INTRODUCTION

In recent years, progress in the development of analysis and optimization tools has made numerical optimization an indispensable component of industrial design processes. In turbomachinery, optimization tools must fully be automated, explore a wide design space and permit the imposition of geometric constraints. Moreover, bringing the geometry back to CAD format and keeping the computational cost as low as possible are both essential.

Recently, gradient–based methods, supported by the continuous or discrete adjoint for computing the objective function gradient, have become very attractive, mainly because the gradient can be computed at a cost independent of the number of design variables. The latter makes the adjoint–based methods suitable for the shape optimization of turbomachinery rows. In CAD–free methods, the objective function sensitivity derivatives w.r.t. the displacement of the nodes lying over the wall boundaries (to be referred to as flow sensitivities), computed by the adjoint solver, are used for modifying the shape of blades and/or casing. In these methods (a) special care must be taken in order to obtain smooth deformed shapes, and (b) the optimized shape must be converted into a CAD format for further processing in the industrial workflow. These issues can be overcome through the use of a parameterization software which is compatible with standard CAD formats. The objective function ($\mathcal{F}$) gradient w.r.t. the design variables $\mathbf{b}$ is computed as

$$\frac{\delta \mathcal{F}}{\delta \mathbf{b}} = \frac{\delta \mathcal{F}}{\delta \mathbf{x}} \frac{\delta \mathbf{x}}{\delta \mathbf{b}} \tag{1}$$

where $\delta\mathcal{F}/\delta\mathbf{x}$ stands for the flow sensitivities computed by the adjoint solver and $\delta\mathbf{x}/\delta\mathbf{b}$ represents the sensitivities of the boundary node positions w.r.t. changes in the design variables (also referred to as geometric sensitivities). In [4], the latter are computed for faceted surfaces by projecting the perturbed geometry onto the original one and applying finite differences. Other approaches rely on the deformation of the surface grids using RBFs followed by a finite difference scheme applied to the resulting and initial grids, [16]. Automatic differentiation of CAD kernels is under investigation by other researchers but only a few basic functions are supported yet.

The adaptation of the initial grid to the updated CAD parameterized boundaries is a crucial component of an automatic optimization loop. This process is often referred as grid morphing. Two main strategies can be employed. The first one exploits the connectivity of the internal grid nodes, applying spring analogy [8] or pseudo–elastic solid techniques[9] while, in the second, each grid node is moved individually by interpolating the surface grid displacement. RBFs [6],[7] are well established tools used in the second approach, due to their robustness and compatibility with any kind of grid connectivity, while preserving elements quality.

In this paper, the optimization of a compressor stator for minimizing viscous losses while penalizing designs which decrease the static pressure rise, is carried out. The blade parameterization software and the computation of geometric sensitivities are presented in Section 2. The continuous adjoint formulation as well as key features of the GPU–enabled flow and adjoint solver are presented in Section 3. Grid morphing is incorporated in the optimization loop (fig. 1) as also described in Section 3. Finally, results of the stator row optimization are discussed in Section 4.

2 BLADE PARAMETERIZATION METHOD

A parameterization/design software for turbomachinery blades is used. The method creates the row geometry by first parameterizing the shape of the blade's mean camber surface and,

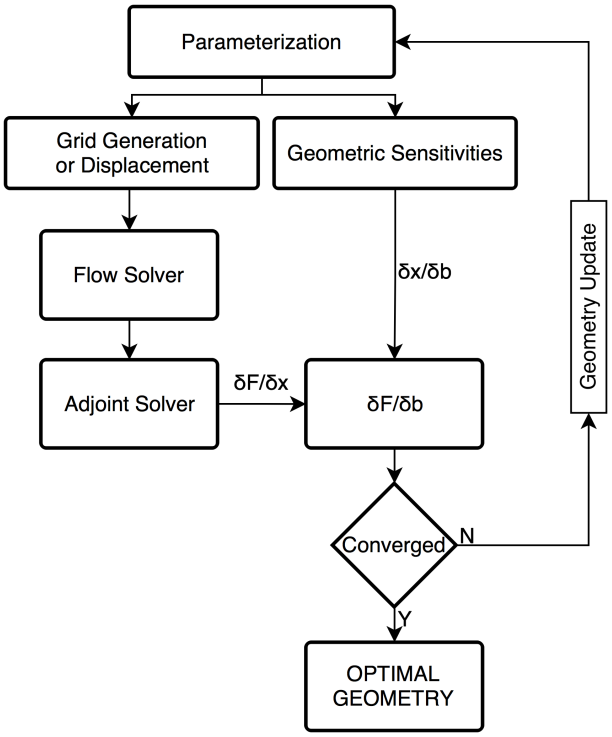

Figure 1: Structure of the overall optimization loop

then, adding thickness.

The majority of data given as input to the parameterization software are Non–Uniform Rational B–Spline (NURBS) curves defining the meridional shape of the row as well as other geometric distributions in the spanwise direction. The NURBS curves provide flexibility to the parameterization method, offering also a compact description for a wide range of blade geometries. In order to parameterize axisymmetric geometries, a conformal mapping [3] of a surface of revolution on the $(m$–meridional, θ–peripheral$)$ plane is employed.

2.1 Outline of the method

The parameterization process and computation of geometric sensitivities $\frac{\delta x}{\delta b_i}$ consists of five basic steps, as follows:

Step 1: Parameterization of the row generatrices and the meridional shape of the LE and TE, Fig. 2.

Step 2: Parameterization of the mean–camber surface of the blade.

Step 3: Superposition of thickness distribution on the mean–camber surface to form the blade shape.

Step 4: Export of the geometry in standard CAD format.

Step 5: Computation of geometric sensitivities.

The first step is the definition of the hub and shroud generatrices via NURBS. In case of a tip clearance between the blade and the hub or shroud, the generatrix representing the meridional shape of the blade tip must also be defined. The meridional shape of the blade LE and TE are

also defined using NURBS curves. Next step is to generate a number of meridional curves, equidistant in the spanwise direction, by interpolating the already defined generatrices for the hub, shroud and blade tip (fig. 2). These curves are used to create surfaces of revolution on which blade sections are defined.

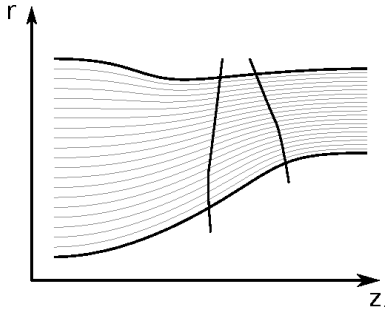

Figure 2: Meridional shape parameterization. Apart from the LE and TE edge meridional projections, the hub and shroud generatrices are shown along with the generatrices of a number of intermediate surfaces of revolution, on which blade sections are defined. Thick lines are defined as NURBS curves, while the thinner ones result from the interpolation of the hub and shroud curves.

After defining the surfaces of revolution for all blade sections, the mean–camber surface is constructed by defining the mean camber line at each section. The mean camber lines are computed on the (m, θ) plane. Since the mapping $(x, y, z) \mapsto (m, \theta)$ is conformal, the blade angles defined on the (m, θ) plane are preserved. The LE and TE of each blade section are defined by their peripheral position (θ_{LE} and θ_{TE}, respectively). Then, the blade metal angles are defined (β_{LE} and β_{TE}, respectively). As shown in fig. 3, the starting and ending–point of the mean camber line ($\mathbf{P}_0$ and $\mathbf{P}_3$, respectively) of each blade section and the point $\tilde{\mathbf{P}}$ where the two tangents intersect are defined. The remaining two control points are located using two non-dimensional weights ζ_{LE} and ζ_{TE}, as follows:

$$\begin{aligned} \mathbf{P_1} &= \zeta_{LE}\mathbf{P_0} + (1 - \zeta_{LE})\tilde{\mathbf{P}} \\ \mathbf{P_2} &= \zeta_{TE}\mathbf{P_3} + (1 - \zeta_{TE})\tilde{\mathbf{P}} \end{aligned} \tag{2}$$

In order to compactly define the θ, β and ζ for the leading and trailing edge and also achieve a smooth geometry, their distributions in the spanwise direction are specified as NURBS curves.

Thicknesses t_{PS} and t_{SS} for the pressure and the suction side, respectively, might be different. To add flexibility they are defined in two steps. First, the normalized thickness ($\hat{t}$) profiles w.r.t. the normalized arc–length of the mean camber line (s) are defined at some spanwise positions. For any other blade section, ($\widetilde{\hat{t}}$) is interpolated. Then, a spanwise distribution for the thickness factor (t_f), that scales the pre-computed profile, is specified. By doing so, the thickness corresponding to a point at distance s along the mean camber line positioned at a normalized spanwise position (η) is given as $t(\eta, s) = \widetilde{\hat{t}}(s, \eta)\ t_f(\eta)$. Having drawn the mean camber line for each blade section on the (m, θ) plane, the normal vector is computed. The two sides of each airfoil are then formed by defining points in the normal direction at a distance specified by the computed thickness distributions scaled as follows

$$\mathbf{c}_{m\theta}(\eta, s) = \boldsymbol{\mu}_{m\theta}(\eta, s) \pm \mathbf{n}_{m\theta}(\eta, s)\frac{t(\eta, s)}{r^2(m(\eta, s))} \tag{3}$$

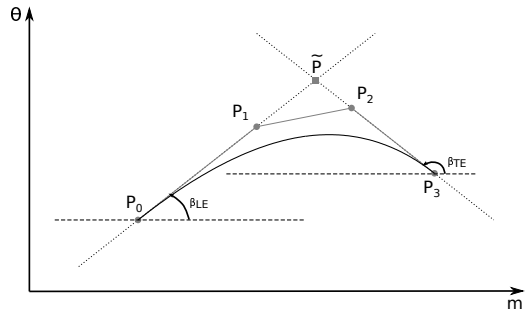

Figure 3: Definition of the mean camber line on the (m, θ) plane. The β_{LE} and β_{TE} angles as well as the curvature of the mean camber line is preserved when mapped back on a surface of revolution, in the 3D Cartesian space.

where $\mathbf{c}$ represents the point on the airfoil in the (m, θ) plane, μ the corresponding point on the mean–camber line and $\mathbf{n}$ the vector normal to the mean–camber line. The resulting blade airfoils on the (m, θ) plane are mapped back onto the 3D Cartesian coordinates to yield the final blade shape.

The parameterization software can export the geometry in neutral CAD format (IGES) for visualization and grid generation purposes. IGES format supports many different entities, among which free–form surfaces in the form of NURBS surfaces that can be used to describe the skin of 3D models.

In the 3D Cartesian space, airfoils are spanwise interpolated through NURBS curves and a skinning algorithm is used to construct the NURBS surface describing the blade; skinning is the process of passing a smooth surface through a set of cross-sectional curves (airfoils). This method requires all cross-sectional NURBS curves to be compatible (equal knot-vector and degree) and can, thus, cause data explosion [5]. Controlling the parameterization of the blade airfoils to have natively compatible cross-sectional curves overcomes this problem.

Hub and shroud surfaces are easily generated as surfaces of revolution based on generatrices and the pitch angle. Similar to the blades themselves, periodic surfaces are generated through skinning. The inlet and outlet boundaries are generated as Coons patches through bilinear blending [15].

Finally, hub and shroud surfaces must be trimmed: in order to avoid numerical instabilities the blade sides are extended. An extension algorithm is applied to the NURBS blade surface [10], as shown in Fig. 4.

Flow sensitivities ($\frac{\delta \mathcal{F}}{\delta \mathbf{x}}$) are computed by the continuous adjoint method for the nodal coordinates of the CFD surface grid. To close the chain rule of eq. 1, geometric sensitivities are needed. Making use of the NURBS representation of the geometry and finite differences, these sensitivities are computed directly on the CFD grid.

Starting from an initial parameterization, the software module perturbs each appropriate design variable and calls the parameterization process, computing the corresponding NURBS surfaces. Surface mesh nodes are inverted onto the NURBS parametric space and, for each surface node, parametric coordinates become available. These parametric coordinates can be used to compute each surface node in the original or any perturbed geometry. Finite differences are applied to the computed nodes. In order to avoid the propagation of numerical errors, intro-

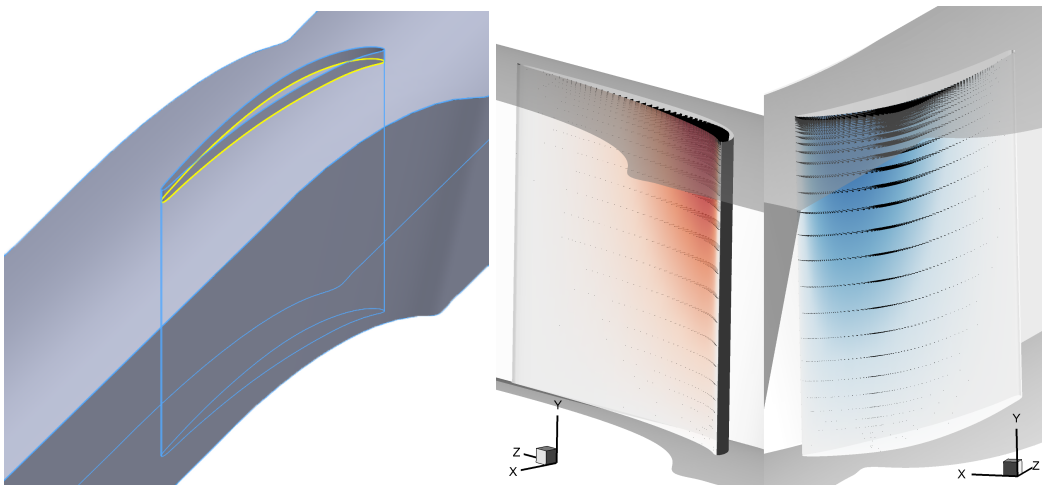

Figure 4: 3D view of the blade extension and the trimming of the blade and shroud surfaces (left). Geometric sensitivities of the blade w.r.t. β_{LE} (fig. 3) at shroud. Contours represent the geometric sensitivities in the normal direction $\frac{\delta \mathbf{x}}{\delta \mathbf{b}}\mathbf{n}$: red denotes inwards (towards the solid) displacement, whereas blue outwards. Arrows represent the actual geometric sensitivities $\frac{\delta \mathbf{x}}{\delta \mathbf{b}}$ (right).

duced by the point inversion algorithm, to the computation of the geometric sensitivities, nodes in the original geometry are recomputed through their parametric coordinates. An application is shown in fig. 4.

3 ADJOINT–BASED SHAPE OPTIMIZATION AND GRID DISPLACEMENT

3.1 Flow Model

The flow model is based on the Navier–Stokes equations for incompressible flows, using the pseudo–compressibility approach introduced by Chorin [1]. By introducing the pseudo–time t the flow equations are expressed as

$$R_n = \frac{\partial U_n}{\partial t} + \frac{\partial f_{nk}^{inv}}{\partial x_k} - \frac{\partial f_{nk}^{vis}}{\partial x_k} = 0 \tag{4}$$

where $n = 1, \ldots, 4$ and $U_n = [p \;\; v_1 \;\; v_2 \;\; v_3]$, with p denoting the static pressure divided by the constant density of the fluid and v_k ($k = 1, \ldots, 3$) the components of the fluid velocity. The inviscid (f_{nk}^{inv}) and viscous (f_{nk}^{vis}) fluxes, in the Cartesian frame, are expressed as

$$f_k^{inv} = \begin{bmatrix} \beta v_k \\ v_k v_1 + p\delta_{1k} \\ v_k v_2 + p\delta_{2k} \\ v_k v_3 + p\delta_{3k} \end{bmatrix}, \quad f_k^{vis} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \end{bmatrix}, \quad \tau_{mk} = (\nu + \nu_t)\left(\frac{\partial v_k}{\partial x_m} + \frac{\partial v_m}{\partial x_k}\right) \tag{5}$$

where β stands for the pseudo–compressibility parameter, ν and ν_t the kinematic and turbulent viscosity respectively, δ_{ij} is the Kronecker symbol and τ_{km} denotes stresses. The turbulent viscosity is computed using the Spalart–Allmaras turbulence model by solving one additional PDE ($R_{\tilde{\nu}}$) for the turbulence variable $\tilde{\nu}$, as in [2].

3.2 Continuous Adjoint for Aerodynamic Shape Optimization

The first objective is to minimize the volume–averaged total pressure (p_t) losses of the flow through a stationary row. Without any loss in generality, this paper is dealing only with stationary bladings, though the programmed software may also handle rotating bladings by switching to a rotating reference frame. The first function to be minimized is expressed as

$$f_1 = -\int_{S_I} p_t v_k n_k dS - \int_{S_O} p_t v_k n_k dS \tag{6}$$

where S_I is the stator inlet and S_O the stator outlet. The second objective, practically, is used to penalize designs which decrease the static pressure rise. This gives the second objective function (being a negative quantity for compressor blading)

$$f_2 = -\int_{S_I} p v_k n_k dS - \int_{S_O} p v_k n_k dS \tag{7}$$

The same method could also be used for the optimization of rotating rows, with the first objective being the minimization of the relative total pressure p_{tR} between inlet and outlet.

The problem, given by eqs. 6 and 7 can be reformulated as a single one by minimizing ,

$$\mathcal{F} = \omega_1 f_1 + \omega_2 f_2 \tag{8}$$

where ω_1 and ω_2 are appropriate positive user–defined weights. To make the cost of computing $\frac{\delta \mathcal{F}}{\delta b_i}$ independent of the number of design variables, the adjoint method is employed. For a function $\mathcal{I}$ (which stands for either f_1 or f_2), the adjoint formulation starts by defining the augmented function

$$\mathcal{I}_{aug} = \mathcal{I} + \int_\Omega \Psi_n R_n d\Omega + \int_\Omega \tilde{\nu}_a R_{\tilde{\nu}} d\Omega \tag{9}$$

where Ψ_n are the adjoint mean flow variable fields and $\tilde{\nu}_a$ the adjoint to turbulence variable $\tilde{\nu}$. By differentiating eq.9 and applying the Leibniz theorem, we have

$$\frac{\delta \mathcal{I}_{aug}}{\delta b_i} = \frac{\delta \mathcal{I}}{\delta b_i} + \int_\Omega \Psi_n \frac{\partial R_n}{\partial b_i} d\Omega + \int_\Omega \tilde{\nu}_a \frac{\partial R_{\tilde{\nu}}}{\partial b_i} d\Omega + \int_S \Psi_n R_n \frac{\delta x_k}{\delta b_i} n_k dS + \int_S \tilde{\nu}_a R_{\tilde{\nu}} \frac{\delta x_k}{\delta b_i} n_k dS \tag{10}$$

Using the Green–Gauss theorem, the volume integrals of eq. 10 are transformed into (a) volume integrals containing $\frac{\partial U_n}{\partial b_i}$ and $\frac{\partial \tilde{\nu}}{\partial b_i}$, ($b$) surface integrals containing $\frac{\delta U_n}{\delta b_i}$ and $\frac{\delta \tilde{\nu}}{\delta b_i}$ and (c) surface integrals containing $\frac{\delta x_k}{\delta b_i}$. Similarly, $\frac{\delta \mathcal{I}}{\delta b_i}$ results in surface integrals containing $\frac{\delta U_n}{\delta b_i}$ and $\frac{\delta \tilde{\nu}}{\delta b_i}$.

By eliminating all integrals containing $\frac{\partial U_n}{\partial b_i}$, [18], the adjoint to eqs. 4, are derived. These are

$$\begin{aligned} &\frac{\partial \Psi_1}{\partial t} \quad -\frac{\partial \Psi_{k+1}}{\partial x_k} = 0 \\ &\frac{\partial \Psi_{m+1}}{\partial t} \quad -v_k \left(\frac{\partial \Psi_{k+1}}{\partial x_m} + \frac{\partial \Psi_{m+1}}{\partial xk} \right) - \frac{\partial \tau_{mk}^{adj}}{\partial xk} - \tilde{\nu} \frac{\partial \tilde{\nu}_a}{\partial x_m} + \mathcal{T}_m^{TM} = 0, \quad m = 1, \ldots, 3 \end{aligned} \tag{11}$$

where the adjoint stresses are given as $\tau_{km}^{adj} = (\nu + \nu_t) \left(\frac{\partial \Psi_{m+1}}{\partial xk} + \frac{\partial \Psi_{k+1}}{\partial x_m} \right)$ and $\mathcal{T}^{TM}$ stands for terms resulting from the differentiation of the turbulence model. Similarly, eliminating volume integrals containing $\frac{\partial \tilde{\nu}}{\partial b_i}$ results to the adjoint to the turbulence model equation, [17]. Eliminating the corresponding surface integrals gives rise to the adjoint boundary conditions: the remaining

terms give the final expression of the gradient. The expression of the gradient of $\mathcal{I}$ w.r.t. the coordinates of all surface nodes is

$$\frac{\delta \mathcal{I}}{\delta x_k} = -\int_{S_w} \beta \Psi_1 n_m \frac{\partial v_m}{\partial x_\ell} \frac{\delta x_\ell}{\delta x_k} dS + \int_{S_w} \Psi_{m+1} n_q \frac{\partial \tau_{qm}}{\partial x_\ell} \frac{\delta x_\ell}{\delta x_k} dS - \int_{S_w} \tau_{mq}^{adj} n_q \frac{\partial v_m}{\partial x_\ell} \frac{\delta x_\ell}{\delta x_k} dS \qquad (12)$$

Since the inlet and outlet boundaries remain unchanged during the optimization, $\frac{\delta x_k}{\delta b_i} = 0$ along S_I and S_O.

3.3 Numerical Solution of the Flow and Adjoint Equations on GPUs

The primal and adjoint solvers use the vertex-centered variant of the finite volume method on unstructured/hybrid meshes. Both solvers are implemented on NVIDIA GPUs using CUDA-C, taking advantage of the great parallel speed–up they offer. Some comments on the discretization and numerical solution techniques used as well as their implementation on GPUs follow.

The primal inviscid fluxes are computed using the Roe's approximate Riemann solver [11] adapted to incompressible flows. Adjoint fluxes are computed in a similar way, by considering the non-conservative form of the field adjoint equations. The spatial discretization of the inviscid fluxes is second–order accurate, with appropriate flux limiters, if necessary. Both primal and adjoint viscous fluxes are computed using an edge–based central difference scheme. The discretized equations are solved in each pseudo-time step using a point–implicit Jacobi iterative scheme.

Modern GPUs are massively parallel co-processors to CPUs offering at least an order of magnitude more FLOPS and higher memory bandwidth than CPUs. However, due to the different architecture, software directly ported from CPUs to GPUs cannot efficiently exploit the available GPU computational resources. The authors group has developed a flow and adjoint solver that is capable of running on many GPUs belonging to the same or different computational nodes with a parallel speed–up of up to 45x compared to the CPU implementation of the same code. This parallel speed–up results from *a*) the optimization of memory access patterns for the used GPU architecture [14], *b*) the use of GPU specific scatter–adding techniques [13] *c*) the minimization of the communication overhead when using many GPUs, by simultaneous data transfers and computations.

Another important feature of the flow and adjoint solver is the use of Mixed–Precision Arithmetics(MPA); the interested reader could find more about MPA in [12]. In order to reduce memory footprint and number of memory accesses, in MPA, the LHS terms are computed in Double–Precision Arithmetics (DPA) and stored using Single–Precision Arithmetics (SPA). Of course, the RHS terms are still stored using double–precision, so as to maintain the accuracy of a purely DPA scheme.

3.4 Grid Displacement using RBF

With the RBF model the deformation is treated as a scattered data interpolation where surface node displacements are smoothly interpolated at the internal nodes. The new surface grid, which corresponds to the new geometry, is obtained by inverting [15] and displacing nodes in the NURBS parametric space (u, v). Special care is taken in case of trimmed surfaces. An application is shown in fig. 5.

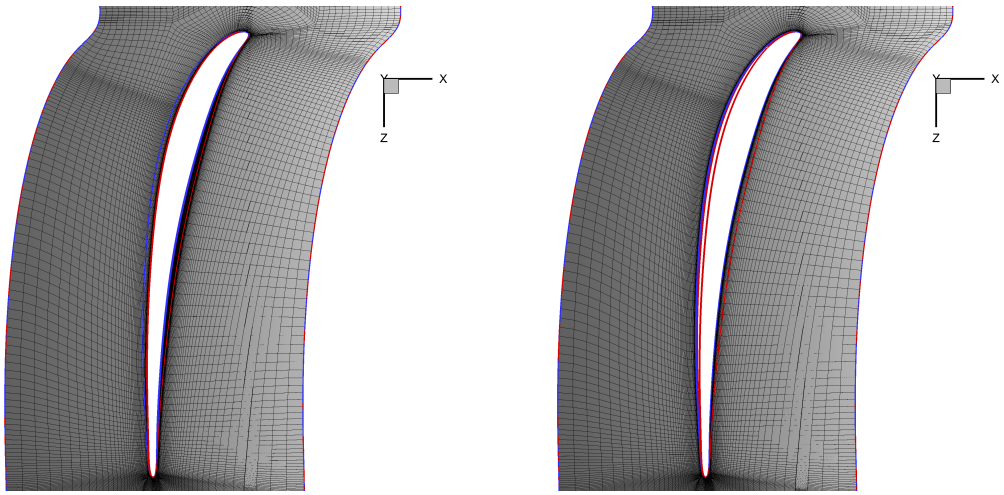

Figure 5: Example of the use of the RBF-based grid displacement method showing the initial (left) and displaced at cycle 4 (right) shroud surface mesh. All boundary nodes are displaced into the NURBS parametric space and, then, transformed back to the Cartesian space.

4 APPLICATION TO A LOW–SPEED COMPRESSOR STATOR

The test case concerns the optimization of the stationary row of a low–speed compressor. The objective is to minimize the volume averaged total pressure losses between the inlet and outlet of the CFD domain penalizing designs which decrease the volume averaged static pressure rise. The weights used in the objective function 8 are $\omega_1 = 0.9$ and $\omega_2 = 0.1$. The inlet flow angle is 46° w.r.t. the axial direction and $Re = 6.5{\times}10^5$ based on the blade axial chord–length (c_{ax}). The initial blade is formed by the spanwise extrusion of the same airfoil and trimming with the cylindrical hub and shroud.

The total number of design variables used to carry out the optimization is 40; note that 32 out of the 40 variables are used to control the blade thickness distribution. In fig. 6, the initial distribution of the thickness is reported along with the NURBS control polygon; control points are allowed to vary only in one direction. In order to maintain $C0$ continuity, at the trailing and leading edge, the first and last control point are kept fixed.

The metal angles β_{LE} and β_{TE} and parameters ζ_{LE} and ζ_{TE} are used to control the blade camber at the hub and shroud positions. Only two span positions are used to describe the distribution of these geometric characteristics. A graphical representation is shown in fig. 6, together with a 3D view of the stator to be optimized. The hub and shroud generatrices remain frozen.

The optimization run on a single computation node with two 6-core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz processors and two NVIDIA Tesla K40 GPUs with 12GB of GPU memory. The computational grid consists of ~$1.9{\times}10^6$ nodes and is carefully stretched close to the solid walls, where the non-dimensional distance of the first grid nodes off the wall is $y^+{\approx}1$. The solution of the flow field runs for ~$17min$ while ~$12min$ are needed for the solution of the adjoint equations per optimization cycle.

The optimization history as well as both the initial and optimized shapes, are shown in fig. 7.

To compare the flow fields around the initial and optimized geometries 4 stations/positions in the axial direction are chosen as shown in fig. 8. In figs. 9 to 10, the flow fields around the initial and the optimized blade geometries are compared.

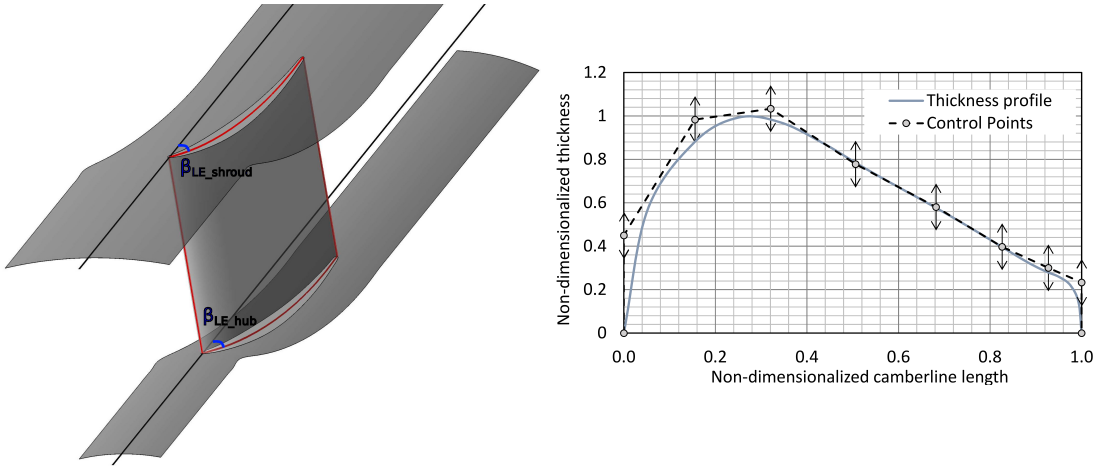

Figure 6: Parameterization of the β angles at LE and TE for hub and shroud position (left). The normalized thickness profile distributions are defined as NURBS curves (right). The control points of the NURBS curves are used as design variables: displacements in the y direction (defining the non–dimensionalized thickness) are safely allowed.

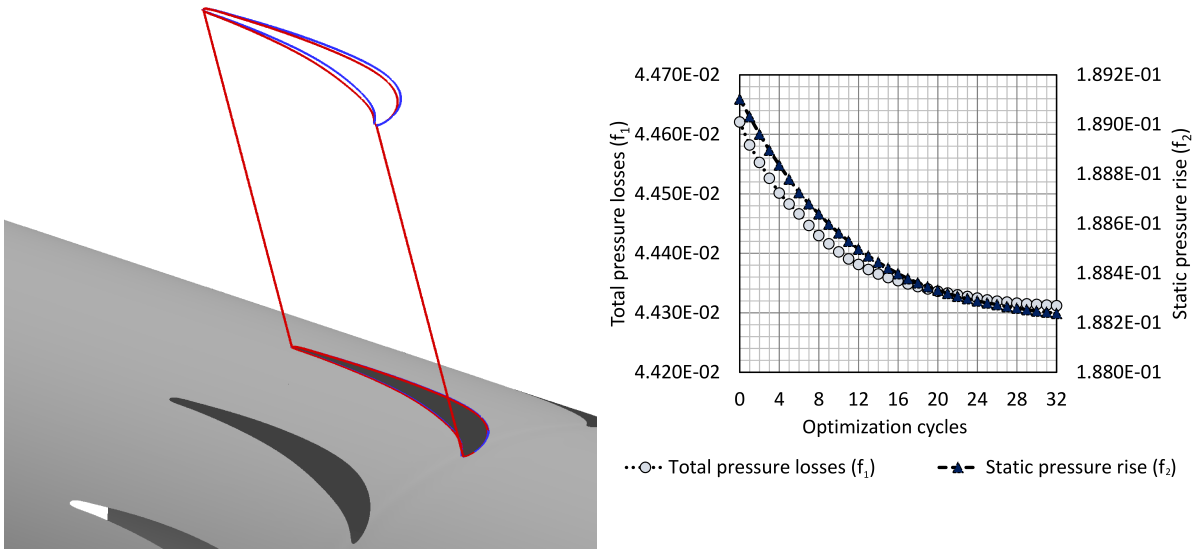

Figure 7: Comparison of the initial (blue) and the optimized (red) blade shape (left). Convergence history for f_1 and f_2 during the optimization (right). Both absolute convergence history (right–top) and relative convergence history (right–bottom) are reported. Both p_t and p are non–dimensionalized by $(V^{in})^2$.

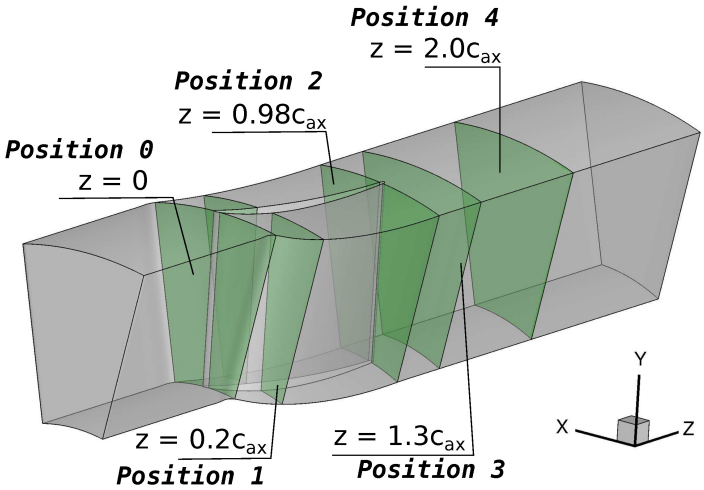

Figure 8: Axial positions chosen to compare the flow fields around the initial and optimized geometry.

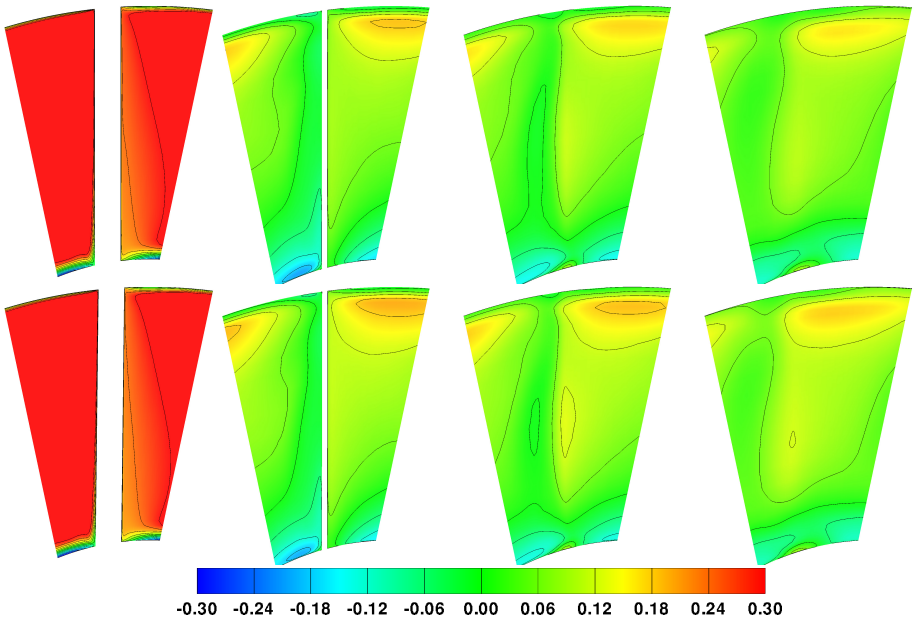

Figure 9: Peripheral velocity component field at axial positions from 1 (left) to 4 (right) for the initial (top) and the optimized (bottom) geometry. The peripheral velocity component is non-dimensionalized by V^{in}.

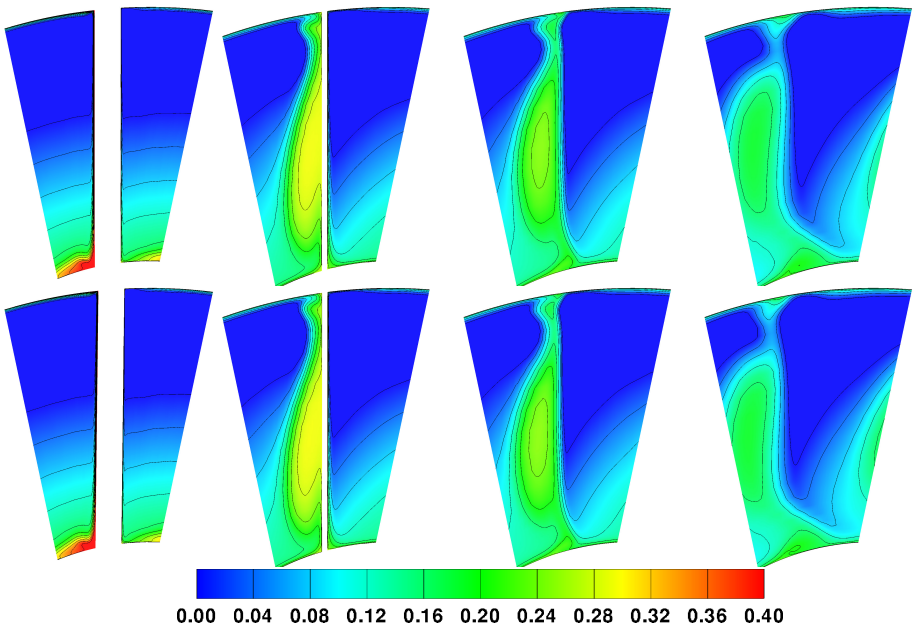

Figure 10: Loss coefficient ($p_t^{in} - p_t$) field at axial positions from 1 (left) to 4 (right) for the initial (top) and the optimized (bottom) geometry. The total pressure is non-dimensionalized by $(V^{in})^2$.

5 CONCLUSIONS

A method for the optimization of turbomachinery rows, by coupling the adjoint–based optimization process with a parametric blade modeller was presented. This enables the exploration of the design space while maintaining a CAD representation. The use of the (continuous) adjoint method makes the cost of computing the objective function gradient independent of the number of design variables. Moreover, the optimization wall–clock time is further reduced by implementing the flow and adjoint solvers on GPUs, speeding–up the corresponding processes by ~45x, compared to the use of CPUs. Regenerating the CFD grid for each new geometry was overcome by using a grid displacement technique based on RBFs. The reduced cost, combined with the capability of linking to CAD, makes the proposed method appropriate for the industrial design workflow.

6 ACKNOWLEDGEMENT

This research was funded from the People Programme (ITN Marie Curie Actions) of the European Union's H2020 Framework Programme (MSCA-ITN-2014-ETN) under REA Grant Agreement no. 642959 (IODA project). The second author is an IODA Early Stage Researcher.

REFERENCES

[1] A. Chorin, A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics,* ***2-1***, 12–26, 1967.

[2] P. Spalart, S. Allmaras, A one–equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale*, **1**, 5–21, 1994.

[3] M. Rossgatterer, B. Jüttler, M. Kapl, G. Della Vecchia, Medial design of blades for hydroelectric turbines and ship propellers. *Computers & Graphics*, **36**, 434–444, 2012.

[4] T.T. Robinson, C. G. Armstrong, H. S. Chua, Carsten Othmer, T. Grahs, Optimizing parameterized CAD geometries using sensitivities based on adjoint functions. *Computer-Aided Design & Applications*, **9(3)**, 363–268, 2012.

[5] L. Piegl, W. Tiller, Algorithm for approximate NURBS skinning. *Computer-Aided Design*, **9(28)**, 699–706, 1996.

[6] M. Buhmann, Radial basis functions: theory and implementations. *Cambridge University Press, Cambridge*, 2003.

[7] D. Sieger, S. Menzel, M. Botsch, High quality mesh morphing using triharmonic radial basis functions. *Proceedings of the 21st International Meshing Roundtable*, 1–15, 2013.

[8] S. Zhu, A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains. *Finite Elements in Analysis and Design*, **41**, 1118–1139, 2005.

[9] Z. Xu, M. Accorsi, Finite element mesh update methods for fluid structure interaction simulations. *Finite Elements in Analysis and Design*, **40**, 1259–1269, 2004.

[10] S.M. Hu, C.L. Tai, S.H. Zhang, An extension algorithm for B-splines by curve unclamping. *Computer Aided Design*, **34**, 415–419, 2002.

[11] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, **43(2)**, 357–372, 1981.

[12] I. C. Kampolis, X. S. Trompoukis, V. G. Asouti, K. C. Giannakoglou, CFD–based analysis and two–level aerodynamic optimization on Graphics Processing Units. *Computer Methods in Applied Mechanics and Engineering*, **199(9-12)**, 712–722, 2010.

[13] V. G. Asouti, X. S. Trompoukis, I. C. Kampolis, K. C. Giannakoglou, Unsteady CFD computations using vertex–centered finite volumes for unstructured grids on Graphics Processing Units. *International Journal for Numerical Methods in Fluids*, **67(2)**, 232–246, 2011.

[14] X. S. Trompoukis, V. G. Asouti, I. C. Kampolis, K. C. Giannakoglou, CUDA implementation of vertex–centered, finite volume CFD methods on unstructured grids with flow control applications. *GPU Computing Gems, Jade Edition*, 2011.

[15] L. Piegl, W. Tiller, The NURBS book. *Springer*, 2013.

[16] A. Ronzheimer, Aircraft geometry parameterization with high-end CAD-software for design optimization. *Proceedings ECCOMAS 2012, Vienna, Austria*, 2012.

[17] E. M. Papoutsis-Kiachagias, K. C. Giannakoglou, Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Archives of Computational Methods in Engineering*. *Springer*, 2016.

[18] K. T. Tsiakas, X. S. Trompoukis, V. G. Asouti, K. C. Giannakoglou, Shape optimization of wind turbine blades using the continous adjoint method and volumetric NURBS on a GPU cluster. *Proceedings of EUROGEN 2015, Glasgow, UK*, 2015.