A Two–Step Mesh Adaptation Tool Based on RBF with application to Turbomachinery Optimization Loops

Flavio Gagliardi, Konstantinos T. Tsiakas and Kyriakos C. Giannakoglou

Abstract Adapting an unstructured CFD mesh to the modified geometry, in accordance with the updated value-set of design parameters at the end of each cycle, is a must in CFD–based shape optimization loops. Mesh adaptation is a nice alternative to remeshing procedures which might become expensive and, also, hinder the initialization of new simulations from previous results. Mesh morphing, based on Radial Basis Functions (RBF) network, has been widely used in the past to smoothly propagate boundary nodal displacements into the volume mesh while preserving its validity and quality. To precisely capture even small design changes, all surface mesh nodes must be used as interpolation nodes which, in case of large meshes for real-world application, leads to excessive computational cost and memory requirements. This paper introduces a cost reduction strategy for mesh adaptation, by proposing a new two-step RBF interpolation employing the Sparse Approximate Inverse (SPAI) preconditioner and the Fast Multipole Method (FMM). The software is demonstrated in the aerodynamic shape optimization of a turbomachinery row. The purpose of this paper is not to solve the optimization problem itself; emphasis is laid on the way the proposed method may handle large displacements and, for this reason, Evolutionary Algorithms (EA) which allow great variations in the values of the design variables were first used. Adjoint-based optimization follows; its role is to perform the refinement of the best solution obtained by the EA-based search.

Flavio Gagliardi, PhD Student,
e-mail: fl.gagliardi@gmail.com
Konstantinos T. Tsiakas, PhD Student,
e-mail: tsiakost@gmail.com
Kyriakos C. Giannakoglou, Professor,
e-mail: kgianna@central.ntua.gr
National Technical University of Athens (NTUA), School of Mechanical Engineering, Parallel CFD & Optimization Unit, Athens, Greece.

1 RBF-based Mesh Displacement: Introduction & Literature Survey

To perform an automated CFD-based aerodynamic shape optimization, a flow solver, a shape parameterization method, an optimization technique and a procedure to adapt or regenerate the CFD mesh for each new candidate solution must be available.

The method presented in this paper focuses onto the problem of CFD mesh adaptation and is demonstrated in the optimization of a compressor stator. In specific, an RBF-based mesh adaptation technique, able to smoothly propagate the displacements of all surface mesh nodes to the interior of the domain has been devised and programmed. This requirement springs from the need of adapting an existing mesh to an updated CAD representation of the shape to be optimized. This can be used in optimizations which employ a CAD system to build the geometry, with the CAD parameters as design variables. In this paper, an in–house parameterization/design software for turbomachinery bladings [18] is used to generate a NURBS-based representation of the geometry.

Obtaining a new surface mesh conforming with the changed boundaries is the starting point for deforming the volume mesh. This paper, however, focuses only on the adaptation of the volume mesh given the displacements of the surface mesh nodes which are obtained by inverting and displacing nodes in the NURBS parametric space, taking special care for trimmed surfaces [18].

RBF-based interpolation methods are robust but may become computationally expensive, especially for large meshes. In section 2.1, it is shown that, by using a data-reduction algorithm, fewer nodes are used to approximate the new shape, reducing dramatically the computational cost and memory requirements compared to the standard formulation. However, this is expected to deteriorate the geometrical precision of boundaries. In section 2.2, a strategy to recover the deviation of the surface mesh with respect to the prescribed shape, caused by the previously made approximation, is proposed.

The theory of RBF networks, [5], is briefly summarized below. RBF networks can *interpolate* discrete data in the n-dimensional space. In mesh displacement, quantities to be interpolated are the known displacements defined at *source nodes* or *RBF centres*. An RBF deformation function $\boldsymbol{d} : \mathbb{R}^3 \to \mathbb{R}^3$ is a linear combination of radially symmetric kernels $\phi_s(\boldsymbol{y}) = \phi(||\boldsymbol{x}_s - \boldsymbol{y}||)$[1] centered at the N *source nodes* $\boldsymbol{x}_s \in \mathbb{R}^3$ and weighted by $\boldsymbol{w}_s \in \mathbb{R}^3$:

$$\boldsymbol{d}(\boldsymbol{y}) = \sum_{s=1}^{N} \boldsymbol{w}_s \phi_s(\boldsymbol{y}) \tag{1}$$

where $\boldsymbol{y}$ is the *target node* position vector. All M (boundary and internal) mesh nodes $\boldsymbol{y}_t,\ t \in [1,\ldots,M]$, for which eq. 1 provides their displacements $\boldsymbol{d}(\boldsymbol{y}_t)$ are considered

[1] $||\ldots||$ is the Euclidean norm.

as *target nodes*. The N boundary mesh nodes with known displacements $\boldsymbol{\delta}_s \in \mathbb{R}^3,\ s \in [1,\ldots,N]$ are used as *source nodes* $\boldsymbol{x}_s$.

Weights $\boldsymbol{w}_s$ are computed so as to exactly reproduce the imposed displacements $\boldsymbol{\delta}_s$ at source nodes; this requires the numerical solution of a linear system with an $N \times N$ symmetric positive-definite[2] coefficient matrix $\boldsymbol{A}$, namely:

$$\begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_N(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_N) & \cdots & \phi_N(\boldsymbol{x}_N) \end{bmatrix} \begin{pmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_N^T \end{pmatrix} = \begin{pmatrix} \boldsymbol{\delta}_1^T \\ \vdots \\ \boldsymbol{\delta}_N^T \end{pmatrix} \tag{2}$$

The computation of weights $\boldsymbol{w}_s$, by solving eq. 2, is the most computationally expensive task. It shows poor scalability if implemented naively, due to both the complexity of linear solvers and its stiffness. After solving eq. 2, the displacements $\boldsymbol{d}(\boldsymbol{y}_t)$ for all mesh nodes $\boldsymbol{y}_t$ are computed by eq. 1 at the cost of $M \times N$ RBF kernel evaluations. The behaviour of the RBF interpolation is highly influenced by the chosen kernel ϕ [4].

Mesh adaptation based on RBF interpolation has being standing out in the literature for their wide range of application. Selim et al. [17] discuss advantages and disadvantages of the most widely used techniques, such as linear and torsional springs, linear elasticity and several interpolation based methods. Based on their work, mesh deformation based on RBF interpolation is one of the most promising approach in terms of robustness and morphed mesh quality, on condition that its high computational cost and bad scalability can be mitigated by methods such as greedy data reduction algorithms.

Greedy algorithms [11] start from a coarse approximation to the deformation and iteratively refine it until the desired accuracy be reached. They use a subset of the surface mesh nodes to describe the new shape, leaving the rest of the nodes for error checking. These methods are more efficient than standard RBF interpolation but they cannot reproduce exactly all surface deformations. The iterative procedure, required to guarantee the error drop to a prescribed tolerance is, for tight surface tolerances, time consuming. Some authors also suggested to apply a correction step such as an explicit interpolation [16] or Delaunay graph mapping [19] after the approximation step, but locally supported RBF interpolation appears to be a better choice from the quality and robustness point of view [10]. Other methods aiming at reducing the RBF-based interpolation cost can be found in the literature. RBF multilevel techniques involve successive levels of nested RBF interpolations where, at each level, the solution from the previous coarser level is interpolated [15, 14, 8]. In this paper, the interpolation is practically carried out on two levels, with the advantage of being able to use other cost reduction methods, which have a dominant setup time and would be impractical to use in many levels. In [13], a multiscale RBF interpolation which uses multiple support radii to capture deformations at different scales is presented. The interpolation matrix is built starting from a coarse subset of source nodes. The algorithm proceeds by iteratively adding the remaining source nodes using a support radius such that the newly added nodes do not affect the

[2] Under certain conditions explained in [5].

previous, ending up with an easier to solve linear system. In such a method, the necessary preprocessing phase cost is dominant, and it is suggested to be performed once before all mesh adaptations.

2 The Proposed Two-Step RBF Mesh Displacement Strategy

The proposed method works by hierarchically using an approximate predictor step followed by a correction one. Both rely on RBF networks. The two steps are briefly described as follows:

- In the first step (*predictor*), all mesh nodes (surface, interior) become interpolation targets and a new coarsened set of source nodes is generated by a non-iterative data reduction method. This method is *adaptive*, in the sense that the data reduction is performed by taking into account the displacement field to be interpolated instead of just the spatial distributions of mesh nodes. The entire mesh is displaced; however, boundary mesh nodes do not precisely respect the known boundary displacements since a reduced number of sources is used.
- The second step (*corrector*), corrects the position of the surface mesh nodes, through local deformations.

The first step generates a ”small” but dense coefficient matrix (its rank might be by orders of magnitude lower than the number of surface mesh nodes) while the second generates a ”big” (rank equal to the number of surface mesh nodes), though very sparse, matrix.

2.1 Step 1: Predictor

The predictor is an RBF approximation tool based on data reduction according to which the source nodes are coarsened by clustering. Cost reduction does not rely only on the reduced problem size but, also, on the implementation of methods such as the Sparse Approximate Inverse (SPAI) preconditioner [12] and the Fast Multipole Method (FMM) [9]. The predictor is divided into three sub-steps: data reduction, training and application, which are described below.

2.1.1 Data Reduction

The objective of the data reduction phase is to find a reduced set of RBF centres $\boldsymbol{x}_r,\ r \in [1,\ldots,N_R]$ which is representative of the displacement field of the surface mesh nodes $\boldsymbol{x}_s,\ s \in [1,\ldots,N_S \gg N_R]$. For this purpose, an *adaptive octree* data structure is employed, which recursively splits the Cartesian space. By taking into account the surface mesh nodes $\boldsymbol{x}_s$ density and the spatial gradient of the displacements $\nabla\boldsymbol{\delta}_s$ the collocation of more RBF centres in areas of rapid variation of the imposed displacements is ensured.

The centres of leaf (childless) octree boxes $\boldsymbol{x}_r$ are used as RBF centres in the predictor training step. The displacement $\boldsymbol{\delta}_r$ of each RBF centre $\boldsymbol{x}_r$ is the averaged displacement of the source nodes $\boldsymbol{x}_s$ contained in the corresponding leaf box of the octree. Such a method does not allow the error to be estimated a priori or reduced iteratively, but it quickly generates the reduced point clouds to approximate the displacement field. This approach is preferred since any error in the reproduction of the imposed displacements will be resolved in the following corrector step. Figure 1 shows an example of selected RBF centres and the corresponding CFD surface mesh.

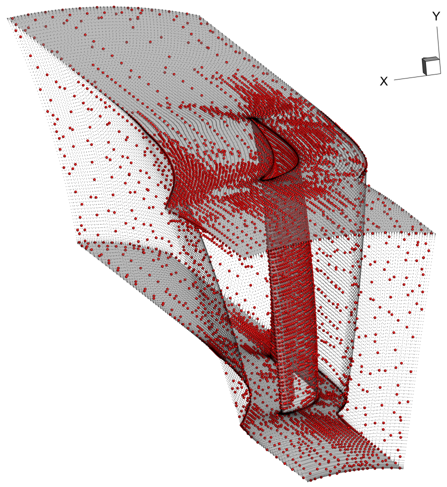

Fig. 1 RBF centres $\boldsymbol{x}_r$ (red spheres) generated by the data reduction algorithm in the predictor step (for a certain displacement of the CFD mesh on the compressor blade). Original mesh surface nodes $\boldsymbol{x}_s$ are displayed as black dots. More RBF centres lie in the area of high spatial gradient of the displacements $\nabla\boldsymbol{\delta}_s$. Generally, the RBF centres $\boldsymbol{x}_r$ do not lie on the mesh surface.

2.1.2 RBF Network Training

The training process runs on the previously generated RBF centres $\boldsymbol{x}_r$. A global support RBF kernel is chosen taking into account different characteristics, such as mesh quality preservation[16], flop count, condition number of the generated linear system and smoothing effect. In this step, the following kernel is used

$$\phi(r) = \frac{1}{\frac{r}{\sigma} + 1} \tag{3}$$

where σ is the shape parameter regulating the width of the kernel and r the euclidean distance between two nodes. The linear system of eq. 2, assembled with $\boldsymbol{x}_r$ as RBF centres, is solved by an iterative method.

A *Sparse Approximate Inverse* (SPAI) [12] preconditioner is implemented to speed-up the convergence of the iterative solver. This preconditioner is, in general, not symmetric, and a solver for non-symmetric matrices i.e., Bi-Conjugate-Gradient-Stabilized (BiCGStab), has to be used . The SPAI preconditioner **M** is an approximate inverse of an approximation to $\boldsymbol{A}$ (eq. 2). The method is based on the minimization of the Frobenius norm

$$\min_{\boldsymbol{M}} ||\boldsymbol{S}\,\boldsymbol{M} - \boldsymbol{I}||_F^2 \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{S}$ is a sparse matrix formed by the largest entries of $\boldsymbol{A}$.

The sparsity pattern of $\boldsymbol{S}$ is defined a priori through a sparsification strategy based on geometric considerations [6], avoiding thresholding strategies: for each RBF centre, all other centres in the neighborhood, from which the biggest entries of $\boldsymbol{A}$ arise, are selected as entries of $\boldsymbol{S}$. Thanks to the decaying RBF kernels, the largest entries of $\boldsymbol{A}$ are arranged in bands and the largest entries of $\boldsymbol{A}^{-1}$ are expected to be at the same location with the largest entries of $\boldsymbol{A}$, [7], so that, for $\boldsymbol{M}$, the same or a similar sparsity pattern to $\boldsymbol{S}$ can be employed. Figure 2 shows the pattern of the large entries of an RBF training matrix $\boldsymbol{A}$ and its inverse ($\boldsymbol{A}^{-1}$), for the CFD mesh of the turbomachinery case.

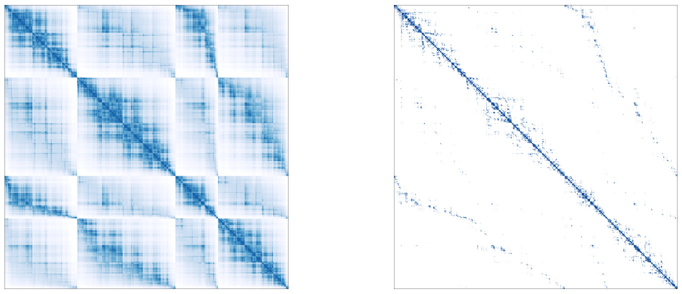

Fig. 2 Pattern of the large entries in a predictor RBF training matrix $\boldsymbol{A}$ (left) and its inverse $\boldsymbol{A}^{-1}$ (right). Large to small entries are shown from blue to white. The matrix rank is $\sim 10^4$ and is originated from the turbomachinery case shown in Figure 1.

In eq. 4, the computation of $\boldsymbol{M}$ is based on a property of the Frobenius norm that allows to split it into a sum of Euclidean norms

$$||\boldsymbol{S}\,\boldsymbol{M} - \boldsymbol{I}||_F^2 = \sum_{i}^{N_R} ||\boldsymbol{S}\,\boldsymbol{m}_i - \boldsymbol{e}_i||_2^2 \tag{5}$$

where $\boldsymbol{m}_i$ and $\boldsymbol{e}_i$ are the i^{th} columns of $\boldsymbol{M}$ and $\boldsymbol{I}$. Each summand in eq. 5 constitutes a linear system, which is solved separately from each other with Cholesky decompositions. The rank of each linear system is noticeably reduced thanks to the sparsity of $\boldsymbol{m}_i$. The number of decompositions needed is also significantly reduced using geometric considerations: in fact, all RBF centres in the same neighborhood, identified by an integer lattice[3], will lead to the same reduced linear system which is decomposed just once, [6]. This procedure overestimates the fixed radius neighbors which leads to bigger linear systems. However, this is compensated by the reduced number of linear systems to be solved, reduced complexity in the neighbors search and higher quality of the preconditioner due to the greater number of entries.

Figure 3 shows the time required for the solver to converge including the setup time for various preconditioners.

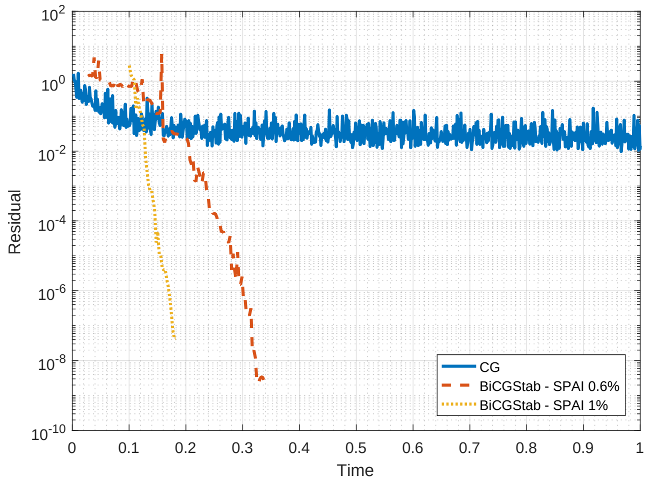

Fig. 3 History of the residual of a full linear system (rank $\sim 4\times10^4$) for various combinations of iterative solvers and SPAI preconditioners plotted as the function of normalized time. In the sake of fairness, the setup time for the preconditioners, which appears as a delay before the solvers take over, is considered too. Percentages in the legend refer to the density (which is equal to 1 minus the sparsity of the matrix) of the preconditioners. The non-preconditioned CG solver does not have any setup time but the convergence rate is badly affected by the system ill-conditioning. Two different SPAI preconditioners were used, with different densities to show that a correlation exists between density and quality but, of course, a denser preconditioner requires more time to be built.

2.1.3 RBF Network Application

After having solved eq. 2 for the weights $\boldsymbol{w}_r$, the interpolated displacement field results from eq. 1 applied at each mesh node. For large meshes the application time can noticeably be reduced using the Fast Multipole Method (FMM) [9]. FMM is an

[3] An integer lattice is tessellation of the $\mathbb{R}^3$ euclidean space in bricks. It is equivalent to a level of an octree.

algorithm for approximating sums such as those appearing in eq. 1, with reduced complexity and controllable error. Briefly, the FMM exploits the decay of the RBF kernel by computing interactions between mesh nodes on different levels of accuracy depending on their geometric distances. This is achieved by low-rank approximations to the displacement field in conjunction with a hierarchical decomposition of the Cartesian space. The quality of the low-rank approximations determines the error made by the method.

There is a trade-off between complexity and approximation error and whether this approach becomes advantageous or not depends on the mesh size but, also, the minimum mesh nodal distance, which determines the maximum allowed multiplication error. In fact, the risk is to introduce a great error (due to the prescribed tolerance) in the interpolated displacements which could yield critical mesh elements quality. Figure 4 shows the time required to perform RBF network applications for increasing mesh sizes with and without the FMM. The FMM-based RBF network application is cheaper for big meshes.

The implementation relies on the black–box FMM (bbFMM)[9]. It is “black-box” in the sense that the functional form of the low–rank approximation is independent of the RBF kernel used since this is based on polynomial interpolation on Chebyshev nodes.

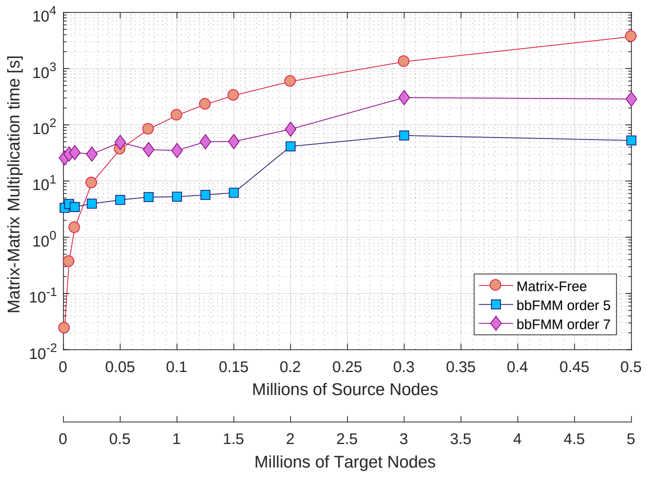

Fig. 4 Wall clock time for the evaluation of eq. 1 by varying the number of source and target nodes (always in the ratio 1:10) using the bbFMM method for the RBF kernel of eq. 3. The FMM-based multiplications include the FMM setup time. Two interpolation orders, 5 and 7, are shown for the bbFMM, introducing a maximum approximation error (infinity norm) smaller then 1×10^{-5} and 1×10^{-7}, respectively. Measurements were performed on a computational node with two 6-core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz processors.

2.2 Step 2: Corrector

The corrector step is based on a local RBF interpolation method. Locality is ensured by the kernel formulation, which vanishes when the distances r of two mesh nodes is higher than the so-called *local support radius* r_s. The locally supported RBF kernel used in this step is the Wendland C0 function [20]:

$$\phi(r) = \begin{cases} \left(1 - \frac{r}{r_s}\right)^2 & \text{if} \quad r < r_s \\ 0 & \text{if} \quad r \geq r_s \end{cases} \tag{6}$$

A tradeoff between the smooth propagation of the deformations in the volume mesh, computation time and memory requirements, depending on the choice of the support radius, is expected. Lower support radius leads to better conditioned and sparser training matrix (see eq. 2) whereas deformation is dissipated in a smaller portion of the interior mesh.

In the corrector, the RBF centres coincide with the surface mesh nodes with prescribed displacements. Since the predictor has already displaced the surface mesh nodes close to their target positions, the remaining surface displacements are relatively small and a small local support radius can be chosen.

The concept of *fixed-radius neighbors search* is used to reduce the matrix filling time in eq. 2. An integer lattice, scaled so that the distance between lattice points is r_s, is built to map nearby to each other mesh nodes. The lattice is, then, used to compute non-zero interactions $\phi(r)$ between close nodes instead of all pairwise interactions.

After solving the sparse linear system, with the help of the SPAI preconditioner, the displacement field is obtained by evaluating eq. 1 at all mesh nodes. By searching in a fixed-radius area, the computation of null kernel values of eq. 1 is avoided.

3 Compressor Stator Blade Optimization Results

The two–step mesh adaptation tool is tested by performing an EA-based, followed by an adjoint-based, optimization of the blade shape of the TU Berlin TurboLab low–speed compressor stator [2]. Inlet conditions are provided in the form of radial profiles, corresponding to 39.7° average inlet flow angle w.r.t. the axial direction, 104 kPa average inlet total pressure and 301 K average inlet total temperature. The outlet static pressure is adjusted to impose 9.5 kg/s full-annulus mass flow rate.

The objective is to minimize the mass-averaged deviation of the exit flow from the axial direction, defined as

$$\alpha = \left(\frac{\int_{S_O} \left(\cos^{-1} \left(\frac{V_a}{|\boldsymbol{V}|} \right) \right)^2 \rho V_a \, \mathrm{d}S}{\int_{S_O} \rho V_a \, \mathrm{d}S} \right)^{\frac{1}{2}} \tag{7}$$

where V_a is the axial component of the velocity, $|\boldsymbol{V}|$ the velocity magnitude, ρ the density and S_O and S_I the stator outlet and inlet sections.

The in–house GPU enabled RANS solver for compressible flows [3], employing the Spalart-Allmaras turbulence model, and its adjoint were used. The blade is parameterized with the in-house turbomachinery row CAD software [18], with 133 design variables. The mesh is block-structured with $\sim 2.2 \times 10^6$ nodes.

In section 3.1, the performance of the mesh adaptation model is discussed. Optimization results follow in section 3.2.

3.1 Mesh Adaptation to the Displaced Boundaries

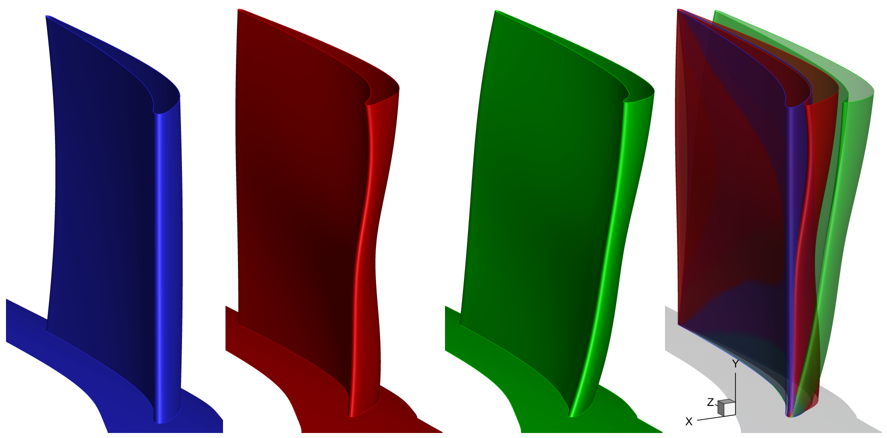

Fig. 5 EA and adjoint-based optimization of a low-subsonic compressor stator blade: reference geometry (1st), geometry generated by the EA–based optimization (2nd) which was used to explore the design space before switching to the adjoint-based optimization (3rd). On the right (4th), the shapes of the three blades are superimposed.

The mesh adaptation tool is tested by displacing the initial compressor stator mesh to the improved design generated by the hybrid optimization. Fig. 5 shows the reference mesh and those resulting from the optimization runs. Table 1 reports quality metrics for the reference and adapted meshes (resulting from the adjoint-based optimization).

	Reference	Adapted
Min. Jacobian	>0	>0
Min. Orthogonality	0.144	0.115
Avg. Orthogonality	0.79	0.76
Max. Normal Skewness	0.856	0.885
Avg. Normal Skewness	0.21	0.24
Max. y^+	0.50	0.51

Table 1 Quality metrics for the reference and adapted block structured volume mesh ($\sim 2.2 \times 10^6$ nodes) in the optimal geometry resulted from the adjoint-based optimization and shown in figure 5. The sign of the Jacobian is used to check the validity of the mesh. Larger orthogonality metric values and lower normal skewness values are desirable to avoid deteriorating the CFD solution accuracy and robustness. Max. $y^+ < 1$ of the first nodes of the wall is required to guarantee that the mesh near the wall is adequate for CFD simulations.

Table 2 tabulates metrics showing the deviation (distance) of the reference and displaced surface meshes at each step of the mesh adaptation procedure. The first step reduces significantly the deviation but the resulting surface mesh does not perfectly fit to the new geometry. This is corrected during the second step.

	Initial	1^{st} Step	2^{nd} Step
Infinity Norm	3.27×10^{-2}	5.30×10^{-4}	4.85×10^{-14}
Euclidean Norm	3.83	1.83×10^{-2}	1.51×10^{-12}
Avg. Deviation	4.87×10^{-3}	2.31×10^{-5}	2.57×10^{-15}

Table 2 Deviation metrics for the reference and displaced CFD surface meshes ($\sim 1.20 \times 10^5$ nodes), resulting from the adjoint-based optimization, figure 5. The first column lists the values for the surface mesh deviation prior to mesh adaptation. Columns labeled "1^{st} and 2^{nd} Step" give the surface mesh deviation upon completion of the corresponding steps.

Figure 6 shows an analysis conducted in order to investigate the time and memory requirements to perform a mesh displacement for growing mesh sizes. The predictor application and corrector training are the most expensive ones. The former scales linearly with mesh size, thanks to the Fast Multipole Method. The latter scales super-linearly due to the BiCGStab computational complexity and increased matrix size and setup time of the SPAI preconditioner. The predictor training time is almost constant since the imposed surface mesh displacements are similar for all mesh sizes. The corrector application phase also scales super-linearly due to the increased number of RBF kernel evaluations, but its contribution to the total required time remains minimal thanks to the computation strategy which is based on the integer lattice.

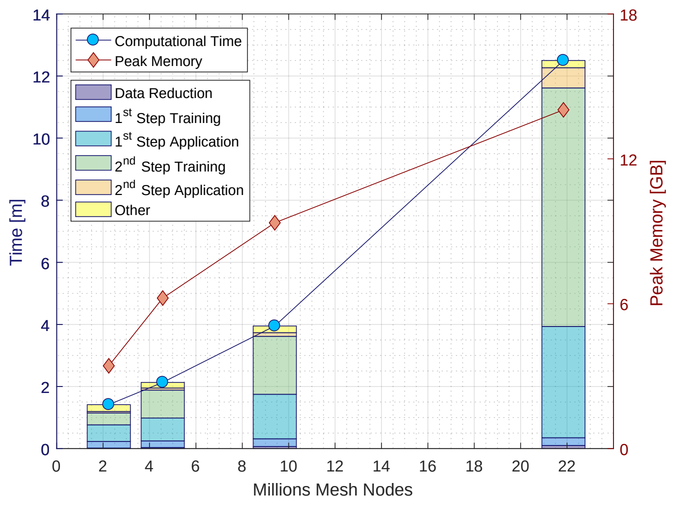

Fig. 6 Computational time and max. RAM memory requirement for mesh displacements, by varying the mesh size. The computational time is broken down in the five main steps in the bar chart: Clustering (§2.1.1), predictor training (§2.1.2) and application (§2.1.3) as well as corrector training and application (§2.2). The bar chart and computational time refer to the left vertical axis, while the peak memory curve points to the right one. Measurements are performed on a computational node with two 6-core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz processors.

3.2 Aerodynamic Shape Optimization Results

The EA-based optimization was performed using the s/w EASY (Evolutionary Algorithm System)[1] developed by the NTUA group. Only the 30 most important design variables of the geometry, reparameterized by the in-house turbomachinery row parameterization software, were used. The adjoint-based optimization relied upon Sequential Quadratic Programming (SQP). The turbomachinery row parameterization software was differentiated and coupled with the in-house GPU-enabled continuous adjoint solver, for computing the gradients [18].

The initial design shows a deviation of the exit flow from the axial direction of 5.52°. The EA-based design space exploration was able to reduce it to 3.98° at the cost of 150 CFD simulations. The adjoint-based optimization resulted to an even better solution with α equal to 3.16°, at the cost of 40 equivalent CFD simulations. The deviation of the exit flow from the axial direction for the reference and optimized blades are shown in Fig.7.

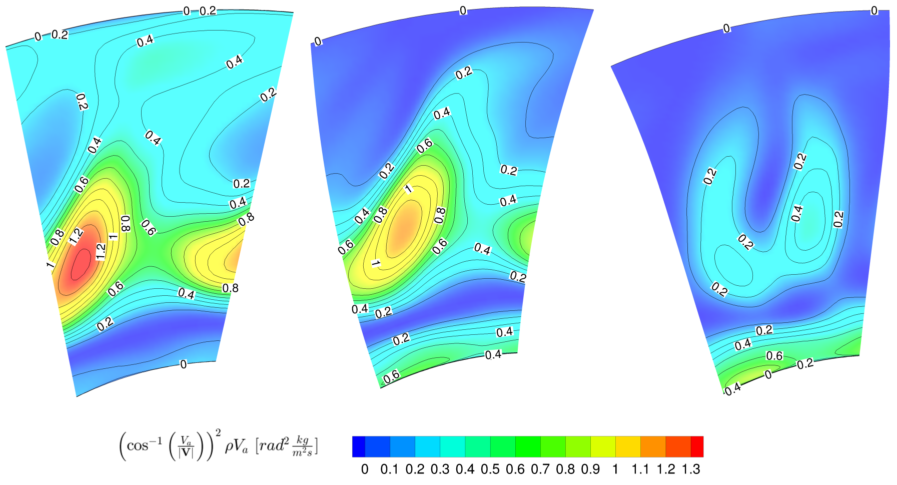

Fig. 7 $\left(\cos^{-1}\left(\frac{V_a}{|\mathbf{V}|}\right)\right)^2 \rho V_a$ field (see eq.7) at the stator outlet for the reference (left), best solution from the EA (centre) and best solution from the adjoint-based optimization(right).

4 Conclusions

An efficient mesh adaptation method, based on RBF, that uses all surface mesh nodes to ensure an exact surface representation was presented. This is achieved by using a correction step, to move all surface mesh nodes to their exact positions, following an approximate step (predictor), taking care of the largest part of the displacements. These steps are furthermore accelerated using the SPAI preconditioner based on geometric considerations and the Fast Multipole Method.

The reliability of the two-step strategy in cases with relatively large displacements and the significant scalability, in terms of computational time and memory requirements for large meshes, have been shown.

During the hybrid optimization of a low-subsonic compressor stator, the proposed mesh adaptation method was successfully used, reducing the computational resources (compared to a re-meshing strategy) required to improve the row design and demonstrating its flexibility in handling large and small displacements. The results show that the proposed two-step mesh adaptation model has high efficiency and its cost scales almost linearly with the mesh size, preserving mesh quality consistently even in large design variations.

Acknowledgements The authors thank Dr. Varvara G. Asouti and Dimitrios H. Kapsoulis for their assistance assistance during the EA-based optimization with the s/w EASY (Evolutionary Algorithm System) [1] and Dr. S. Xenofon Trompoukis for his assistance with the in-house flow and adjoint solvers [3].

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 first author is an IODA Early Stage Researcher.

References

[1] (2008) The EASY (Evolutionary Algorithms SYstem) software, http://velos0.ltt.mech.ntua.gr/EASY

[2] (2016) AboutFlow Project Website: TU Berlin TurboLab Stator Case, http://aboutflow.sems.qmul.ac.uk/events/munich2016/benchmark/testcase3/

[3] Asouti VG, Trompoukis XS, Kampolis IC, Giannakoglou KC (2011) 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

[4] de Boer A, van der Schoot M, Bijl H (2007) Mesh deformation based on Radial Basis Function interpolation. Computers and Structures 85(1114):784 – 795

[5] Buhmann M (2009) Radial Basis Functions: Theory and Implementations. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press

[6] Carpentieri B (2009) Algebraic preconditioners for the fast multipole method in electromagnetic scattering analysis from large structures: trends and problems. Electronic Journal of Boundary Elements 7(1)

[7] Demko S, Moss WF, Smith PW (1984) Decay rates for inverses of band matrices. Mathematics of computation 43(168):491–499

[8] Floater MS, Iske A (1996) Multistep scattered data interpolation using compactly supported radial basis functions. Journal of Computational and Applied Mathematics 73(1):65 – 78

[9] Fong W, Darve E (2009) The black-box fast multipole method. Journal of Computational Physics 228(23):8712 – 8725

[10] Gillebaart T, Blom D, van Zuijlen A, Bijl H (2016) Adaptive radial basis function mesh deformation using data reduction. Journal of Computational Physics 321:997–1025

[11] Hon Y, Schaback R, Zhou X (2003) An adaptive greedy algorithm for solving large RBF collocation problems. Numerical Algorithms 32(1):13–25

[12] Kallischko A (2007) Modified sparse approximate inverses (MSPAI) for parallel preconditioning. PhD thesis, Technische Universitat Munchen, Germany

[13] Kedward L, Allen CB, Rendall T (2017) Efficient and exact mesh deformation using multiscale RBF interpolation. Journal of Computational Physics 345:732 – 751

[14] Lazzaro D, Montefusco LB (2002) Radial basis functions for the multivariate interpolation of large scattered data sets. Journal of Computational and Applied Mathematics 140(1):521 – 536

[15] Narcowich FJ, Schaback R, Ward JD (1999) Multilevel interpolation and approximation. Applied and Computational Harmonic Analysis 7(3):243–261

[16] Rendall TCS, Allen CB (2010) Parallel efficient mesh motion using radial basis functions with application to multi-bladed rotors. International Journal for Numerical Methods in Engineering 81(1):89–105

[17] Selim M, Koomullil R (2016) Mesh deformation approaches–A survey. J Phys Math 7(181):2090–0902

[18] Tsiakas KT, Gagliardi F, Trompoukis XS, Giannakoglou KC (2016) Shape optimization of turbomachinery rows using a parametric blade modeller and the continuous adjoint method running on GPUs. In: 7th ECCOMAS Conference Proceedings, Crete Island, Greece

[19] Wang Y, Qin N, Zhao N (2015) Delaunay graph and radial basis function for fast quality mesh deformation. Journal of Computational Physics 294:149 – 172

[20] Wendland H (1995) Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Advances in computational Mathematics 4(1):389–396