A Two–Step Radial Basis Function-Based CFD Mesh Displacement Tool

Flavio Gagliardi (flavio@central.ntua.gr)
Kyriakos C. Giannakoglou (kgianna@central.ntua.gr)

National Technical University of Athens (NTUA), School of Mechanical Engineering, Parallel CFD & Optimization Unit, Athens, Greece.

Abstract

Mesh displacement based on Radial Basis Functions (RBF) interpolation is known for its ability to preserve the validity and quality of the mesh, even for large displacements, without being affected by mesh connectivity. However, in the case of large meshes, such as those used in real-world Computational Fluid Dynamics (CFD) applications, RBF interpolation, in its standard formulation, becomes excessively expensive. This paper proposes a cost reduction technique for mesh displacement based on RBF, by splitting the process into two steps. In the first step, named predictor, a data reduction algorithm that adaptively agglomerates mesh boundary nodes by reducing the RBF interpolation problem size is used. Upon completion of the first step, due to the agglomeration and the fact that the RBF interpolation is applied to the boundary nodes too, the so-displaced boundaries do not match the given displacements; thus, the position of the boundary nodes must be corrected during the second step, named corrector. The latter performs a local deformation based on RBF kernels with local support, to make the boundary conform to the known displacements of its nodes. The proposed method is accelerated by employing the Sparse Approximate Inverse preconditioner based on geometrical considerations and the Fast Multipole Method. The method and the programmed software are validated on three test cases related to the deformation of CFD meshes inside a duct and a turbine stator row as well as around a car model.

1 Introduction

In CFD, the need for adapting an existing mesh to displaced boundaries arises in many applications including aerodynamic shape optimizations, aeroelastic simulations and flow simulations in the presence of moving bodies. Mesh displacement is an alternative to remeshing since the latter might hinder the continuation of new simulations from available numerical solutions on the unstructured mesh, for instance, of the previous domain.

Various mesh displacement methods have been proposed in the past to meet specific requirements springing from diverse types of simulations or even disciplines. These can be classified in several ways. Some of the encountered classifications are methods based on interpolations, control meshes and physical analogies [1], algebraic vs. partial differential equation methods [2], connectivity-based vs. point-based methods [3], methods for structured or unstructured meshes [4] and methods that can more or less efficiently be parallelized [5]. Recent surveys, such as [1] and [6], enumerate the pros and cons of mesh displacement methods based on various applications and quality criteria.

Mesh displacement methods that have been around for a long time are spring analogies that model the mesh as a network of linear [7], torsional [8], semi-torsional [9] and ball-vertex springs [10] and solve the static equilibrium equations to find the updated nodal locations. These methods are efficient for meshes for structural analysis; [1] reported invalid elements and high computational cost to handle large displacements of CFD meshes with stretched elements for viscous flow simulations. Besides, these methods require the connectivity of the CFD mesh to be available, and extension to generic polyhedral hybrid meshes is, thus, difficult.

Continuum elastic approaches have been proposed by several authors (i.e. [11–13]). They displace the mesh by solving the linear elasticity equations on the mesh itself, thus connectivity should be known. Laplacian methods [14] solve the Laplace PDEs to diffuse the surface mesh node displacements into the domain. The method is efficient for single frequency deformations, although large multiple frequency deformations may lead to invalid meshes. Better quality of the displaced mesh can be achieved by solving the biharmonic smoothing equation [15], at increased computational cost. The algebraic damping method [16] is based on the displacement of each internal mesh node in terms of the displacement of the

Preprint submitted to Advances in Engineering Software February 2019
https://doi.org/10.1016/j.advengsoft.2018.11.011

closest node on the moving boundaries. The resulting deformation appears to be very rigid close to the boundaries but, for large mesh deformations, algebraic smoothing might be necessary to improve mesh quality. The Delaunay graph method [17] is based on the generation of a control mesh based on the Delaunay triangulation of the boundary nodes and the mapping of the internal nodes on the Delaunay graph. The triangulation is adapted to geometry changes, and volume mesh nodes are relocated through barycentric interpolation. The Inverse Distance Weighting (IDW) method [18] computes the location of internal nodes through the direct interpolation of the boundary nodal displacements using weights depending on their distance from the boundary. Transfinite interpolation [19] is based on the interpolation of the deformations along mesh lines which is computationally efficient but limited to structured meshes [20].

Mesh displacement based on RBF interpolation has proved to be robust for large deformations [21]. [1] compared the most common techniques, including linear and torsional springs, linear elasticity and several interpolation-based methods such as the RBF and IDW. The paper concluded by recognizing mesh displacement based on RBF interpolation as one of the most promising approaches regarding robustness and mesh quality; its high computational cost and bad scalability can be mitigated by greedy data reduction algorithms. [22] benchmarked a mesh displacement approach based on RBF interpolation against six techniques previously tested in [6] and concluded that the former, though more expensive, yields deformed meshes of better quality. [18] compared mesh displacement based on RBF and IDW and demonstrated a reduction of the computational cost for the latter, compared to the former, by a factor of 20, for a hexahedral mesh for inviscid flow simulations with $\sim 75,000$ cells. However, the RBF method produced slightly better mesh quality than IDW.

The inefficiency of the RBF-based mesh displacement, in its standard form, is due to the need to solve a linear system of equations with rank equal to the number of the displaced and fixed boundary nodes, besides the computation of the displacements of the internal nodes. In recent years, many researchers focused on the cost reduction of RBF interpolation. Despite the progress made in this field during the last years, the problem is still open to new strategies and improvements.

The use of RBF kernels with local support was the first breakthrough to reduce the cost of RBF interpolations [23]. They lead to sparse matrices that can be solved more efficiently. However, a trade-off between the smooth propagation of the deformation (mesh quality) and the sparsity of the matrix (computational cost) exists. For large deformations, local support must be enlarged, and the problem becomes similar to one with RBF kernels with global support, vanishing the benefits of local support [23]. This problem was alleviated in [24] by dividing the deformation into smaller steps controlled by locally supported RBF interpolations with small radii, leading to a series of very sparse linear systems to be solved. Similar methods, named multilevel RBF techniques, involve successive levels of nested RBF interpolations through which, on each level, the solution from the previous coarser level is interpolated [25, 26]. The mesh displacement method proposed in [27] was based on domain decomposition and local RBF interpolations resulting in a series of small problems.

Greedy algorithms [4] typically start from a coarse approximation to the mesh deformation and iteratively refine it until the desired accuracy is reached. Greedy methods use a subset of the surface mesh nodes to describe the new shape, leaving the rest of the nodes for error checking; so, they are more efficient than standard RBF interpolation, although they cannot precisely reproduce all surface deformations. The iterative procedure required to guarantee the error drop to a prescribed tolerance is time-consuming for tight tolerances. [28] compared various data reduction methods both by considering the actually imposed displacements and computing the subset of surface mesh nodes a priori. [3] proposed an agglomeration strategy of the boundary nodes as typically adopted in multigrid methods. [29] proposed an incremental least-squares solver which, similarly to greedy algorithms, uses a subset of the surface mesh nodes to approximate the deformation. [30] proposed the use of a multiscale RBF interpolation that employs multiple support radii to capture deformations at different scales. 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 previous, ending up with an easier to solve linear system. [4] suggested a correction step such as a Delaunay graph mapping, after the approximation step; however, locally supported RBF interpolation appears to be a better choice from the quality and robustness point of view [31].

This paper introduces a two-step cost reduction technique for mesh displacement, Section 3. The two steps successively deform the mesh based on RBF interpolation, the principles of which are outlined in Section 2. The interpolation is further accelerated by employing the Sparse Approximate Inverse (SPAI) preconditioner, Section 4.1, the Fast Multipole Method (FMM), Section 4.2, and an integer lattice-based

method, Section 4.3.

The proposed strategy and the programmed software are used on three benchmark test cases, Section 5. In Section 5.1, the mesh quality resulting from the proposed method is compared with standard RBF. In Section 5.2, the scalability of the software is analyzed in order to make it efficient in cases with huge mesh sizes. In Section 5.3, the performance of the software is investigated by varying the main input parameters. Finally, in Section 5.4, the effectiveness of the proposed mesh displacement method to large displacements is tested by using it in an evolutionary algorithm based optimization.

2 Background of RBF-based Interpolation

An RBF network is a weighted linear combination of RBF kernels interpolating scattered data in the Q-dimensional space. In mesh displacement, in specific, the quantities to be interpolated are the known 3D displacements at the K surface mesh nodes or, generally, at K distinct source nodes. A 3D RBF kernel $\phi_k(\boldsymbol{x}) = \phi(r = \|\boldsymbol{x} - \boldsymbol{x}_k\|)$ is a real-valued function depending on the distance r of a point $\boldsymbol{x} \in \mathbb{R}^3$ from the so-called RBF interpolation source $\boldsymbol{x}_k \in \mathbb{R}^3$. $\|.\|$ stands for the Euclidean distance.

To interpolate the given displacements $\boldsymbol{\delta}_k$, $1 \leq k \leq K$, K RBF kernels centered at the respective nodes must be used. The RBF interpolant $\boldsymbol{d} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ takes the form:

$$\boldsymbol{d}(\boldsymbol{x}) = \sum_{k=1}^{K} \boldsymbol{w}_k \phi_k(\boldsymbol{x}), \tag{1}$$

where the weights $\boldsymbol{w}_k \in \mathbb{R}^3$ are computed so as to exactly reproduce the known displacements $\boldsymbol{d}(\boldsymbol{x}_k) = \boldsymbol{\delta}_k$ at the K source nodes; this requires the numerical solution of a $K \times K$ linear system, with different right-hand side (r.h.s.) arrays.

Equation 1 is often modified by adding a polynomial term to preserve affine motion, i.e. translation, rotation and scaling. In 3D, the RBF interpolant, including a degree-one polynomial, takes the following form:

$$\boldsymbol{d}_\pi(\boldsymbol{x}) = \sum_{k=1}^{K} \boldsymbol{w}_k \phi_k(\boldsymbol{x}) + \boldsymbol{a}_0 + \boldsymbol{a}_1 x + \boldsymbol{a}_2 y + \boldsymbol{a}_3 z, \tag{2}$$

where $\boldsymbol{a}_q \in \mathbb{R}^3,\ 0 \leq q \leq 3$ are the polynomial coefficients. To compute the new degrees of freedom, new conditions are introduced; assuming that the sources are not co-planar these conditions are:

$$\sum_{k=1}^{K} \boldsymbol{w}_k = 0 \quad \text{and} \quad \sum_{k=1}^{K} \boldsymbol{w}_k \odot \boldsymbol{x}_k = \boldsymbol{0}\,, \tag{3}$$

where $\boldsymbol{0} \in \mathbb{R}^3$ denotes the zero vector and $\odot$ the entry-wise product operator. Then, if

$$\begin{gathered} \boldsymbol{\Phi} = \begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_K(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_K) & \cdots & \phi_K(\boldsymbol{x}_K) \end{bmatrix}, \quad \boldsymbol{P} = \begin{bmatrix} 1 & \boldsymbol{x}_1^T \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_K^T \end{bmatrix}, \\ \boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_K^T \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_0^T \\ \vdots \\ \boldsymbol{a}_3^T \end{bmatrix}, \quad \boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\delta}_1^T \\ \vdots \\ \boldsymbol{\delta}_K^T \end{bmatrix}. \end{gathered} \tag{4}$$

the $(K + 4) \times (K + 4)$ linear system to be solved is

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{pmatrix} \boldsymbol{W} \\ \boldsymbol{A} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Delta} \\ \boldsymbol{0} \end{pmatrix}. \tag{5}$$

For large K values, the computation of $\boldsymbol{W}$ and $\boldsymbol{A}$ by solving equation 5 becomes expensive. It exhibits poor scalability if implemented naively, due to both the complexity of linear solvers and its stiffness. Solving equation 5 is referred as the training phase of the interpolation whereas computing the displacements $\boldsymbol{d}_\pi(\boldsymbol{x})$ for all mesh nodes or targets, by equation 2, is the interpolation phase. More about the theory on the solvability of these types of interpolations can be found in [32].

The behavior of the RBF interpolation is profoundly influenced by the chosen kernel ϕ [23], of either local or global support. The former requires the definition of a local support radius r_s which determines the region of influence of the kernel around each source node. Since $\phi(r) \neq 0$ if and only if $r < r_s$, displacements imposed on the source nodes affect only mesh nodes lying inside the region of influence of the corresponding kernel. A low-valued support radius leads to a better conditioned and sparser matrix $\mathbf{\Phi}$ whereas deformation is dissipated over a smaller portion of the interior mesh.

By appropriately selecting the kernel $\phi(r)$, see Section 3, matrix $\mathbf{\Phi}$ becomes symmetric and positive definite. However, the block-matrix on the left-hand side of equation 5 is generally not positive definite.

3 The Two-Step Strategy

The proposed two-step strategy divides the mesh displacement problem into two successive RBF-based interpolation sub-problems:

- In the first step (predictor), all mesh nodes (surface and interior) are targets of a global RBF interpolation, and a coarsened set of source nodes is generated by a data reduction method. The latter takes both the spatial distributions of mesh nodes and the displacement field to be interpolated into account. In this step, the interpolant takes the form of equation 2 that includes degree-one polynomial terms. After displacing the entire mesh though, the boundary nodes do not precisely respect the known displacements.

- The second step (corrector) corrects the position of the surface mesh nodes through local deformations. All surface mesh nodes become sources and only the internal nodes in a small volume close to the surface become targets. In this step, the RBF interpolant takes the form of equation 1.

The two sub-problems are more manageable than the original problem. In fact, the first step generates a "small" but dense coefficient matrix (its rank might be by orders of magnitude lower than the number of surface nodes) whereas the second step generates a "big" (rank equal to the number of surface nodes), though very sparse, matrix. This strategy allows for a noticeable reduction in the computational cost for displacing a mesh.

3.1 Step 1: Predictor

The predictor is based on data reduction according to which the source nodes set is coarsened by clustering. The objective is to generate a reduced set of sources that are representative of the displacement field of the surface mesh. For this purpose, an adaptive octree data structure that recursively splits the Cartesian space is employed. By considering the surface mesh nodal density and the spatial gradient of the displacements, more sources are generated in areas of rapid variation in the imposed surface nodes displacements. In detail, the algorithm starts from a single parent box containing all surface mesh nodes. Each parent box is recursively split based on the number of contained surface nodes (to ensure proper resolution in densely populated zones) and the maximum difference in the displacements of the contained surface nodes (to ensure proper resolution where the displacement field rapidly changes). Empty boxes are ignored and parent boxes, the division of which would yield only one child box, are subdivided irrespective of the above criteria until a maximum depth limit is reached. The centers of leaf (childless) octree boxes are used as RBF interpolation sources in the predictor training phase. Each source takes on the averaged displacement of the surface mesh nodes contained in the corresponding leaf box of the octree. Figure 1 illustrates an example of selected interpolation sources on the CFD surface mesh of a duct.

The approximation to the displacement introduced by the predictor is measured by the nodal error which is defined at each (i^{th}) surface mesh node $\boldsymbol{x}_i$, as:

$$E_i = \sqrt{\Delta \boldsymbol{x}_i^T \Delta \boldsymbol{x}_i}\,, \tag{6}$$

where, in the predictor step, $\Delta \boldsymbol{x}_i = \boldsymbol{\delta}_i - \boldsymbol{d}_{\pi,\text{Predictor}}(\boldsymbol{x}_i)$ is the difference between the known displacement $\boldsymbol{\delta}_i$ of the i^{th} surface mesh node and that computed within the predictor step, $\boldsymbol{d}_{\pi,\text{Predictor}}(\boldsymbol{x}_i)$. In contrast to greedy methods that usually minimize the surface error norm (summing up all surface nodal errors), in the proposed method, any deviation from the known boundary displacements is resolved in the corrector

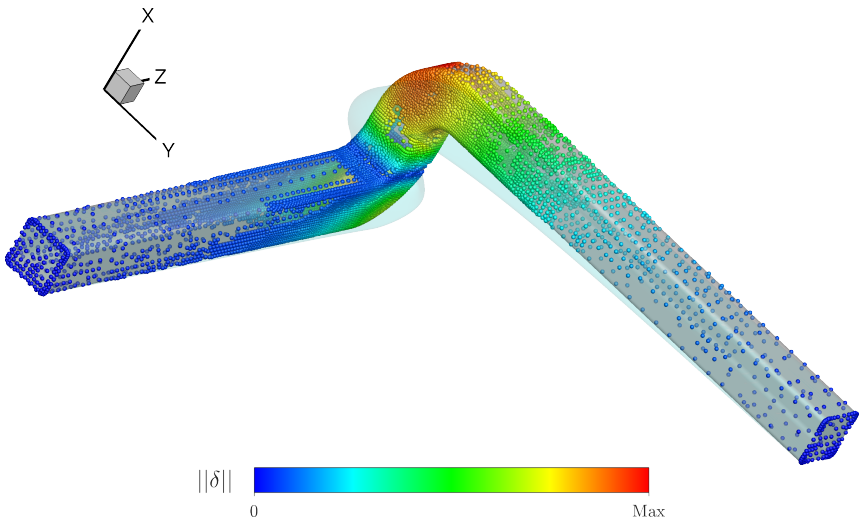

Figure 1: Double elbow duct. RBF sources (spheres) generated by the data reduction algorithm in the predictor step for the duct studied in Section 5. Colors from blue to red represent small to large displacements $\boldsymbol{\delta}$. The reference and displaced duct shapes are depicted in gray and blue, respectively (see also Figure 9). Most RBF sources lie in the area of high spatial gradient of the displacements. The computed sources do not necessarily lie on the mesh surface.

step. The role of the predictor step is to generate a reduced set of source points and approximate the displacement field.

A global support RBF kernel is chosen, by considering characteristics such as mesh quality preservation [4], flop count and condition number of the linear system to be solved. In the predictor step, the inverse multi-quadric kernel [33]

$$\phi(r) = \frac{1}{\sqrt{(\frac{r}{\sigma})^2 + 1}} \tag{7}$$

is employed, where σ is the parameter regulating the decay of the kernel, selected to be equal to half of the mesh bounding box diagonal length.

The linear systems (one per Cartesian direction) in equation 5 (including the polynomial term), assembled with the reduced set of RBF sources, are solved by an iterative method. Since the coefficient matrix in equation 5 is non-positive definite, the selected solver is based on the Bi-Conjugate-Gradient-Stabilized (BiCGStab) algorithm, coupled with the Sparse Approximate Inverse (SPAI) preconditioner. As illustrated in Section 4.1, the SPAI preconditioner is non-symmetric. The iterative method solves systems with multiple r.h.s. arrays (the displacements along the three Cartesian directions) at once using parallel matrix-matrix multiplications. After solving equation 5, the displacement field is obtained by evaluating equation 2 at all mesh nodes. The FMM, Section 4.2, is used to speed up this evaluation.

3.2 Step 2: Corrector

The corrector step is based on a local RBF interpolation method. The kernel used in this step is the Wendland C0 function [34]:

$$\phi(r) = \begin{cases} \left(1 - \frac{r}{r_s}\right)^2 & \text{if} \quad r < r_s \\ 0 & \text{if} \quad r \geq r_s \end{cases}. \tag{8}$$

A tradeoff among the smooth propagation of the deformations in the volume mesh, computational cost and memory requirements, depending on the choice of the support radius, is expected. In the corrector, the interpolation sources coincide with the surface mesh nodes with prescribed displacements. Since the predictor has already displaced the surface mesh nodes close to their known target positions, the remaining surface displacements $\Delta \boldsymbol{x}_i$ are minor. As a rule of the thumb, the support radius should be at least three times larger than the largest error E_i of all surface nodes.

The linear system in equation 5 (without the polynomial term, this time) is solved by taking into account the new nodal positions computed by the predictor and imposing the displacements on the surface nodes to be equal to the already computed differences $\Delta \boldsymbol{x}_i$. The system is sparse and positive definite for this activation function, equation 8; however, since the non-symmetric SPAI preconditioner is employed, the BiCGStab algorithm is used. Then, the displacement field is obtained by evaluating equation 1. The method presented in Section 4.3 is used to speed up this evaluation.

4 Acceleration Methods for the Two-Step Strategy

Methods to efficiently carry out the training and interpolation phases, in both the predictor and corrector steps, are proposed. These methods are:

- The Sparse Approximate Inverse preconditioner [35] for the acceleration of the training phase in both steps.
- The Fast Multipole Method [36] for the acceleration of the interpolation phase in the predictor step.
- An integer lattice-based technique for the acceleration of the interpolation phase in the corrector step.

4.1 The SPAI Preconditioner

Preconditioning is essential to quickly solve the linear system in equation 5, during the training phase in both steps. Recently, preconditioning techniques based on the SPAI have been developed [37]. Their main advantage is that they are inherently parallel since many independent small linear systems must be solved. SPAI preconditioners work properly for a wide range of applications [38] and are immune to numerical difficulties such as pivot breakdowns as well as instabilities which might occur in incomplete LU (ILU) [39].

In the literature, SPAI preconditioners are applied to sparse linear systems, although a few applications with dense systems can also be found. For instance, [40] compared diagonal, symmetric successive overrelaxation, ILU and SPAI preconditioners coupled with a Generalized Minimal Residual (GMRES) solver and found that the SPAI preconditioner gives better convergence rates, for a dense matrix arising from the discretization of an electromagnetic scattering problem.

The SPAI preconditioner $\boldsymbol{M}$ is a sparse approximate inverse of a sparse approximation to $\boldsymbol{\Phi}$ (equation 4). The method assumes that a sparse matrix can effectively approximate the inverse of a full (predictor) or sparse (corrector step) matrix. Recall that the inverse of a sparse and, certainly, a dense matrix is generally dense. Nevertheless, for a decaying RBF kernel, many of the entries in $\boldsymbol{\Phi}$ are small, and many of the entries in $\boldsymbol{\Phi}^{-1}$ are also expected to be small [41]. All these small entries are neglected, so that sparse (or sparser, if $\boldsymbol{\Phi}$ is already sparse) approximations to $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}^{-1}$ are employed.

The computation of the preconditioner $\boldsymbol{M}$ is based on the minimization of the Frobenius norm:

$$\min_{\boldsymbol{M}} \|\boldsymbol{S}\boldsymbol{M} - \boldsymbol{I}\|_F^2 \,, \tag{9}$$

where $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{S}$ is a sparse matrix approximating $\boldsymbol{\Phi}$; $\boldsymbol{S}$ is formed by the largest entries in $\boldsymbol{\Phi}$. In equation 9, the Frobenius norm $\|.\|_F$ is employed since it allows decomposing the minimization problem into K (rank of $\boldsymbol{\Phi}$) independent linear problems. In fact, a property of the Frobenius norm allows splitting it into a sum of Euclidean norms [42]:

$$\|\boldsymbol{S}\boldsymbol{M} - \boldsymbol{I}\|_F^2 = \sum_{k=1}^{K} \|\boldsymbol{S}\boldsymbol{m}_k - \boldsymbol{e}_k\|_2^2 \,, \tag{10}$$

where $\boldsymbol{m}_k$ is the k^{th} column of $\boldsymbol{M}$, and $\boldsymbol{e}_k$ is the k^{th} row of $\boldsymbol{I}$. Each summand in equation 10 constitutes a linear system to be solved, the rank of which is reduced based on the sparsity of $\boldsymbol{S}$ and $\boldsymbol{M}$. For this reason, $\boldsymbol{S}$ and $\boldsymbol{M}$ are subject to sparsity constraints to balance the quality of the preconditioner (the preconditioned system should converge rapidly) and its construction and application time. A complete overview of the theory of the SPAI preconditioner is provided in [35].

The criteria for selecting the sparsity (non-zero) pattern of $\boldsymbol{M}$ are discussed below. The strategy is to maintain low the number of non-zero entries in $\boldsymbol{M}$ while capturing the largest entries in $\boldsymbol{\Phi}^{-1}$ that are

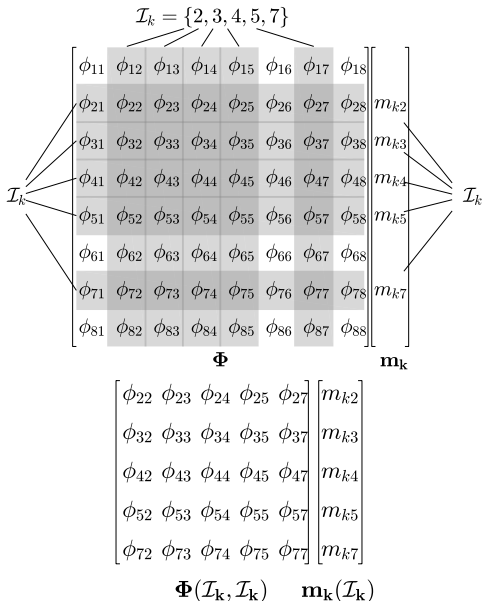

Figure 2: Graphical representation of the definition of the sub-matrix $\mathbf{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ to solve the linear system of each (k^{th}) summand of equation 10. The integer lattice determines the set of indices $\mathcal{I}_k$ (see example in Figure 3). Top: full matrix $\mathbf{\Phi}$ and sparse k^{th} column $\boldsymbol{m}_k$ of the preconditioner $\boldsymbol{M}$. Bottom: the reduced matrix $\mathbf{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ that is solved to find $\boldsymbol{m}_k(\mathcal{I}_k)$.

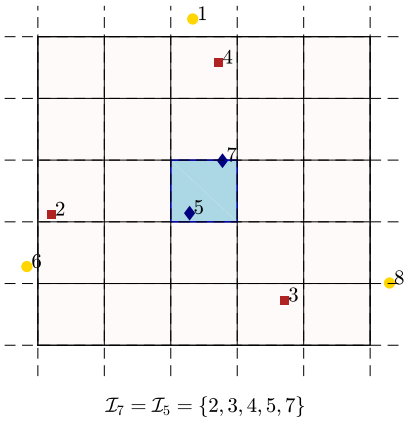

Figure 3: Regular Cartesian 2D integer lattice built over a cloud of sources. Square- and diamond-sources are the third-level lattice neighbors of the diamond-sources. Round marks are considered as far-away sources and, therefore, excluded from the neighbors of the diamond-sources. The integer lattice is used to define a priori the sparsity pattern of the SPAI preconditioner, i.e. sets of indices $\mathcal{I}_k$. The two diamond sources yield the same set of indices $\mathcal{I}_k$. In 3D, the lattice is formed by regular Cartesian cubes.

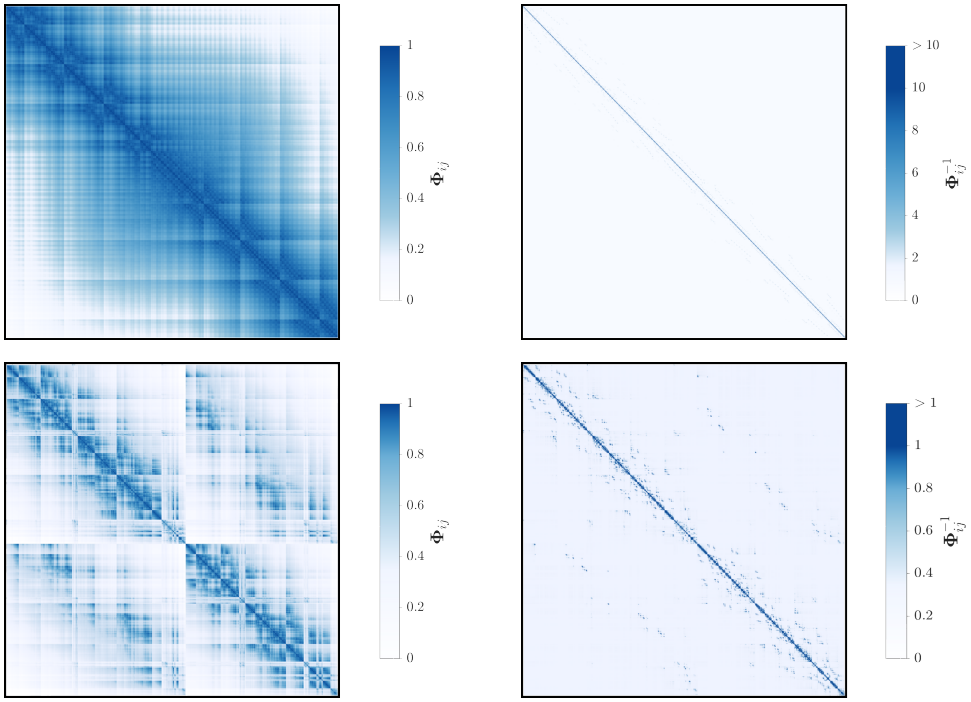

Figure 4: Double elbow duct (top) and turbine stator (bottom). Patterns of two RBF training matrix $\boldsymbol{\Phi}$ (left) and their inverse $\boldsymbol{\Phi}^{-1}$ (right) of the predictor, respectively. Top: $\boldsymbol{\Phi}$ originated from the reduced set of sources of the duct case of Figure 1. Bottom: $\boldsymbol{\Phi}$ originated from the reduced set of sources of the reference shape of the turbine stator case of Figure 11. All matrices are symmetric, and their rank is $\sim 10^4$. Large to small entries are depicted in blue to white. The range $(0, 1]$ of the entries of $\boldsymbol{\Phi}$ corresponds to the image of the kernel in equation 7. Top: in $\boldsymbol{\Phi}^{-1}$, only entries close the diagonal (but not just the diagonal itself) are predominant, since the largest entries in $\boldsymbol{\Phi}$ are close to the diagonal. Bottom: in $\boldsymbol{\Phi}^{-1}$, predominant entries are placed also far form the diagonal and those also match the predominant entries of the inverse.

the most significant regarding the quality of the preconditioner. Reliable results can be obtained using a precomputed sparsity pattern [43]. Thus, strategies that try to dynamically identify a suitable sparsity structure for $\boldsymbol{M}$ are not used. They have general applicability but are time-consuming, and the procedure to identify the preconditioner sparsity structure cannot easily be parallelized [35, 39, 44].

By first assuming that identical sparsity patterns for $\boldsymbol{M}$ and $\boldsymbol{S}$ are given, it is possible to define a set of indices pinpointing the non-zero entries in each column of the preconditioner $\boldsymbol{m}_k$, as follows:

$$\mathcal{I}_k = \{\forall j \in [1, K] \text{ s.t. } \boldsymbol{m}_k(j) \neq 0\}\,. \tag{11}$$

Then, $\mathcal{I}_k$ identifies the columns of $\boldsymbol{\Phi}$ to be kept in the linear system of the k^{th} summand of equation 10. $\mathcal{I}_k$ also identifies the rows of $\boldsymbol{\Phi}$ that are kept, in order to reduce the number of rows of each linear system in such summands. The retained non-zero entries in each column of the preconditioner $\boldsymbol{m}_k$ are computed by solving the following K linear systems:

$$\boldsymbol{\Phi}(\mathcal{I}_k, \mathcal{I}_k)\boldsymbol{m}_i(\mathcal{I}_k) = \boldsymbol{e}_i(\mathcal{I}_k). \tag{12}$$

in which the coefficient matrices are square, symmetric and positive definite. Figure 2 illustrates graphically how the sub-matrix $\boldsymbol{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ is formed by a known set of indices $\mathcal{I}_k$ (as in the example of Figure 3) to compute the entries of the k^{th} column of the preconditioner. With a sparser matrix $\boldsymbol{M}$, smaller sub-matrices $\boldsymbol{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$ are derived so that Cholesky factorization can efficiently solve the linear systems of equation 12, to compute $\boldsymbol{m}_k$. In this regard, if the indices of the rows of $\boldsymbol{\Phi}$ forming the sub-matrices in equation 12 were not the same with the indices of the columns, QR decompositions would be needed being about four times more costly than Cholesky.

Using the same sparsity pattern for $\boldsymbol{S}$ and $\boldsymbol{M}$ is justified by the fact that the position of the large entries in $\boldsymbol{\Phi}^{-1}$, which contribute the most to the quality of the preconditioner, tend to be at the position

of the large entries in $\mathbf{\Phi}$. This is theoretically supported by the work presented in [41]: for a banded positive definite matrix, the entries in its inverse decay exponentially away from the bands. This behavior is illustrated for two RBF coefficient matrices in Figure 4.

The description of the strategy used to define a priori the sparsity pattern of $\boldsymbol{S}$ and $\boldsymbol{M}$, namely the sets of indices $\mathcal{I}_k$, $1 \leq k \leq K$, previously assumed to be given, follows. The sparsity pattern is the sets of indices $\mathcal{I}_k$ identifying the entries with the highest value in each row of $\mathbf{\Phi}$. In fact, for each RBF interpolation source, the largest entries in $\mathbf{\Phi}$ arise from the other sources in the neighborhood, due to the behavior of the RBF kernel. The sets $\mathcal{I}_k$ are, then, formed by the indices of the rows generated by the neighbor sources, including the k^{th} source itself. To facilitate the neighbors' searching procedure, a regular Cartesian integer lattice tessellating the space is constructed. The neighbors of each RBF source are defined through the lattice neighbors. Several levels of lattice neighbors can be defined. First-level neighbors are all nodes in the box containing the source itself. Second-level ones are nodes contained in the neighboring boxes of the first-level box in addition to the first-level itself and so forth. In Figure 3, a 2D explicative lattice is illustrated with three levels of neighbors. The distance between lattice points and the number of levels that define the neighbor's nodes are used to adjust the sparsity pattern of $\boldsymbol{S}$ and $\boldsymbol{M}$.

Using the integer lattice to compute a priori the sparsity pattern has a beneficial effect. In fact, all nodes in a lattice box have the same set of neighbor nodes $\mathcal{I}_k$. The same set of neighbor nodes $\mathcal{I}_k$ leads to the same reduced matrix $\mathbf{\Phi}(\mathcal{I}_k, \mathcal{I}_k)$, which is factorized only once to build all the columns of the preconditioner corresponding to the nodes contained in the lattice box having the $\mathcal{I}_k$ set of neighbors; this reduces significantly the number of Cholesky decompositions needed.

Figure 5 illustrates the time required for the linear solver to converge, in the training phase of the predictor step of the duct case (Section 5.1), including the setup time for various preconditioners. The SPAI preconditioner reduces the time needed to converge to a reasonably accurate solution by one order of magnitude. This saving in time becomes higher for bigger matrices. The influence of the neighbor grouping strategy based on the integer lattice is also illustrated: for the preconditioner with density 5%, the grouping strategy reduces the number of decompositions needed from $\sim 10^4$ to just $\sim 7 \times 10^2$, reducing the set-up time by a factor of approximately 15. As previously mentioned, the SPAI preconditioner is non-symmetric and a solver for non-symmetric matrices (BiCGStab) must be used. Since $\mathbf{\Phi}$ is a symmetric positive definite matrix, a symmetric positive definite preconditioner was also investigated to employ faster iterative solvers. One of them was Conjugate Gradient (CG), used in the corrector step and not in the predictor, due to the polynomial terms which make the coefficient matrix non-positive definite. The Factorized SPAI (FSPAI) preconditioner [35] (positive definite) would exhibit much better performances compared to the SPAI preconditioner if the strategy to reduce the number of sub-matrices decompositions was not included. Since a sub-matrix must be factorized for each column of the preconditioner (factorizations cannot be re-used to build multiple columns), the setup time of the FSPAI becomes predominant compared to the time for solving the system, similarly to the "SPAI column by column" in Figure 5. On the other hand, methods for the symmetrization of SPAI can be used in conjunction with solvers for symmetric non-positive definite matrices. For instance, [45] suggested a symmetrization strategy in conjunction with the Symmetric Quasi-Minimal Residual (SQMR) method, finding better convergence rates that non-symmetric SPAI used in combination with GMRES.

4.2 The Fast Multipole Method

The Fast Multipole Method (FMM) is used to speed up the interpolation phase of the predictor step. Its role is to accelerate the computation of summations involved in equation 1; polynomial terms, which appear in equation 2, are considered separately. The FMM was initially developed in [46] to approximately solve N-body gravitational and electrostatic potential simulations, within any user-defined precision, with runtime complexity $O(N)$ instead of $O(N^2)$ of the direct computation. The same algorithm has been applied to RBF interpolations to compute interactions of RBF kernels [47], thus reducing the computational complexity of the RBF interpolation. Since the FMM is described in detail in the literature (i.e. [48]), its theory or any discussion on parallelization issues are omitted.

The FMM computes the displacement of each target node considering the real contribution of the nearby source mesh nodes and low-rank approximations for the remaining far-away contributions. Therefore, the FMM tool comprises of:

- An octree data structure, used to spatially organize the RBF sources and targets as well as to

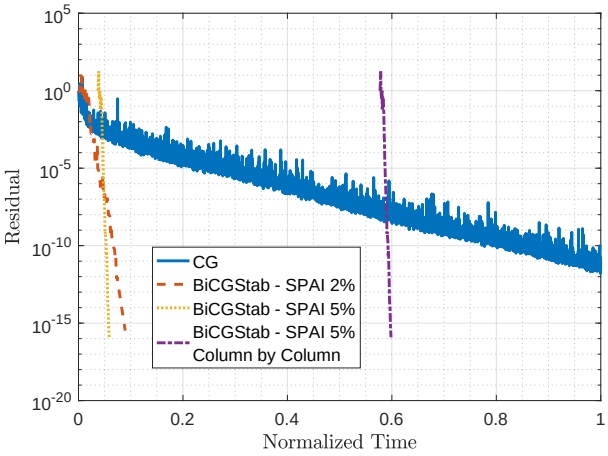

Figure 5: Double elbow duct. History of the residual of the full linear system (rank $\sim 10^4$) assembled from the reduced set of RBF sources (Figure 1) without polynomial terms for various combinations of iterative solvers and SPAI preconditioners plotted as a function of normalized time. For the sake of fairness, the setup time for the preconditioners, which appears as a delay before the solvers take over, is also considered. 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 Conjugate Gradient (CG) solver does not have any setup time but the convergence rate is severely affected by ill-conditioning. Two different SPAI preconditioners were used with different densities, to demonstrate that a correlation exists between density and quality but, of course, a denser preconditioner costs more. The preconditioner named "Column by Column" is built without the neighbors grouping strategy based on the integer lattice: the preconditioners built with the grouping strategy have similar performance and much lower setup time.

differentiate near and far nodes at various levels. The octree data structure adopted in this work is able to deal with highly unbalanced node distributions which are typical in CFD meshes. It is inspired by [48] and implemented as a recursive domain decomposition with parallel divide-and-conquer techniques.

- A low-rank approximation to compute the far-field contributions. In this work, the black-box FMM (bbFMM) [36] is adopted; according to this, the low-rank approximation is based on polynomial interpolation on Chebyshev nodes implemented to compute displacements along the three Cartesian directions simultaneously.

There is a trade-off between computational complexity and approximation error and whether this approach becomes advantageous depends not only on the mesh size but, also, on the minimum nodal distance which determines the maximum allowed error due to the approximations made by FMM. In fact, the risk is to introduce a significant error in the interpolated displacements that could damage mesh quality. Table 1 presents the error introduced in the RBF interpolation by the FMM for two different bbFMM interpolation orders (accuracies). Figure 6 illustrates the time required to perform RBF interpolations for various mesh sizes with and without the FMM. The FMM-based RBF interpolation is more affordable for large meshes even if the parallel implementation is more involved since it requires many synchronization points [49], compared to that of the direct computation. In fact, the direct computation is not affected by parallel slowdown effects, because of the absence of synchronization overheads (the summation for each target node, equation 1, can be carried out independently).

The matrix-matrix products required by the BiCGStab iterative solver applied to equation 5 were replaced by the FMM (the FMM approximates the $\boldsymbol{\Phi W}$ product, whereas $\boldsymbol{PA}$ and $\boldsymbol{P^T W}$ are considered separately) to take advantage of the lowered multiplication complexity. However, issues regarding the accuracy and convergence of the BiCGStab solver arose: nearly exact matrix-matrix products were required for the solver not to converge to a wrong solution, [47]. The approach proposed in [50], based on nested GMRES solvers employing FMM-based matrix-matrix multiplication approximations with different accuracies, was investigated to remedy the issues of the BiCGStab solver using the FMM-based matrix-matrix multiplication. In this nested scheme, the inner solver was used as a preconditioner for the outer solver. The inner solver was using FMM-based matrix-matrix multiplications with reduced

Table 1: RBF interpolation errors and time using the bbFMM for two source and target distributions (distr.) and two FMM interpolation orders. The number of sources is 1.25×10^5 and that of targets 2.2×10^6. The "uniform" distribution of sources and targets is randomly generated in a unit cube whereas in the "CFD" distribution sources and targets correspond to the surface and volume nodes, respectively, of the CFD mesh within the turbine stator case (Section 5.4). Error_∞ and Error_2 are the maximum and average relative errors, respectively. Time appears as the fraction of the time needed to perform the interpolation using the bbFMM over the time needed for the standard RBF interpolation (evaluation of equation 1). Non-uniform distributions (i.e. CFD meshes) require unbalanced octrees to keep the error predictable. However, unbalanced octrees have more levels and complex interaction lists resulting in an increased overall runtime.

		3^{rd} Order			7^{th} Order		
Kernel	Distr.	Error_∞	Error_2	Time	Error_∞	Error_2	Time
$\frac{1}{\sqrt{r^2+1}}$	Uniform	3.77×10^{-3}	2.53×10^{-6}	6.28×10^{-3}	5.14×10^{-8}	1.22×10^{-11}	6.13×10^{-2}
	CFD	4.37×10^{-3}	1.53×10^{-5}	2.71×10^{-2}	1.54×10^{-7}	1.54×10^{-10}	1.35×10^{-1}

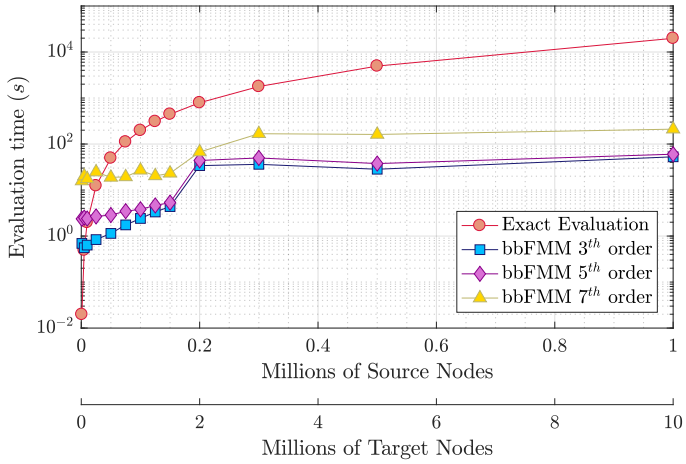

Figure 6: Wall-clock time for the evaluation of equation 1 by varying the number of sources and target nodes (retaining the ratio 1:10) for a uniform distribution randomly generated in a unit cube using the bbFMM method for the RBF kernel of equation 7. The FMM-based interpolations include the FMM setup time. Three interpolation orders, 3, 5 and 7, are illustrated for the bbFMM, introducing a maximum approximation error (relative infinity norm) smaller than 1×10^{-3}, 1×10^{-5} and 1×10^{-7}, respectively.

accuracy (low cost) whereas the outer solver was using accurate ones (at higher cost). The inner solver was preconditioned by SPAI. Even if this approach worked well, the cost has increased, concluding that such an approach seems attractive only for huge dense linear systems which is not the case in either step of the proposed method.

4.3 Integer Lattice-Based RBF Interpolation

The FMM is not used in the corrector since the piecewise definition of the locally supported RBF kernel makes the low-rank approximation by the bbFMM unreliable. Instead, a faster method is proposed for exact interpolation of RBF networks with small local support radii. A neighbors' search procedure based on an integer lattice is used to accelerate the interpolation phase of the corrector step by avoiding the computation of zero-valued summands in equation 1. The integer lattice used in this section is similar to the one explained in Section 4.1 (see also Figure 3) but contains both sources and targets and is scaled so that the distance between lattice points be equal to the local support radius r_s of the RBF network. Then, the contributions from the sources on the first level lattice neighbors are the only ones that must be evaluated to compute the displacement of the target nodes in each lattice box. In fact, due to the

Table 2: Double elbow duct. Quality metrics for the reference and displaced meshes, using standard RBF and the proposed two-step strategy. Lower aspect ratios are favorable regarding CFD solution accuracy.

	Reference	Standard RBF	Two-Step Strategy
Max. aspect ratio	6574	6810	6597
Avg. aspect ratio	4.12	4.02	4.02
Min. cell volume	2.87×10^{-11}	8.99×10^{-12}	1.45×10^{-11}

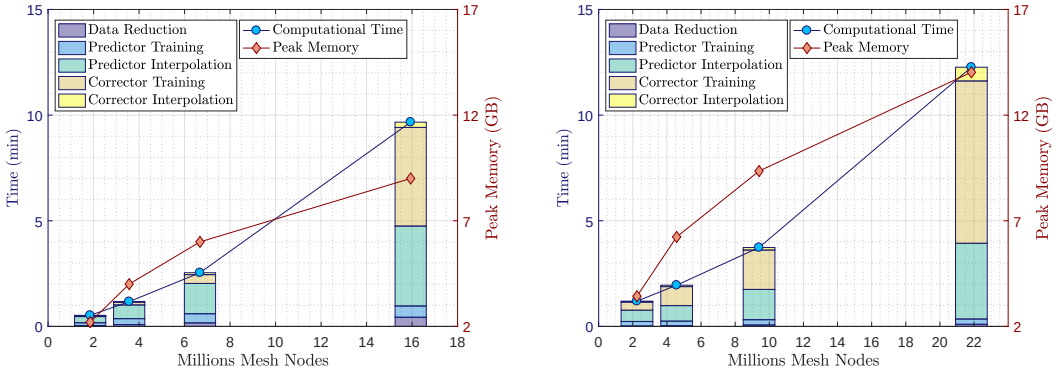

Figure 7: Wall-clock time and RAM requirements for the double elbow duct (left) and turbine stator (right) mesh displacements, for different mesh sizes. Cost is broken down into five main steps in the bar chart: data reduction, predictor training and interpolation (Section 3.1) as well as corrector training and interpolation (Section 3.2). The bar chart and computational time refer to the left vertical axis whereas the peak memory curve points to the right one.

local support of the RBF kernel, only sources within a radius r_s from a target node contribute to its displacement.

This strategy reduces the interpolation cost by about two orders of magnitude for the duct and turbine stator cases presented in Section 5. Moreover, this procedure is readily amenable to parallelization since the computed displacements are independent of each other. Additionally, the same neighbors' search procedure is used to reduce the coefficient matrix filling time in equation 5.

5 Parametric Studies and Results

Table 3: Double elbow duct. Maximum and average surface error norm E_i for the reference and displaced CFD surface mesh illustrated in Figures 1 and 9. The first column lists the values of the error function prior to mesh displacement. The column labeled "Predictor Step" gives the same norms upon completion of this step. The corrector merely drops both error norms to zero.

	Reference	Predictor Step
Max. E_i	2.01×10^{-1}	1.77×10^{-3}
Avg. E_i	5.76×10^{-2}	7.68×10^{-5}

Below, four validation tests, on three 3D CFD meshes, are reported. In Section 5.1, a comparison of mesh displacement performed with the standard RBF model and the proposed method is presented for a double elbow duct. The same double elbow duct and a turbine stator are used in Section 5.2 to assess the software performance and scalability with respect to mesh sizes. In Section 5.3, the performance of the software while varying the predictor step size is investigated for the displacement of a polyhedral mesh, with up to 22-face polyhedral cells. Finally, in Section 5.4, the effect of increasing deformations on the mesh quality of the turbine stator CFD mesh is investigated in the frame of an evolutionary algorithm-based optimization. In all cases, mesh quality assessment is based on commonly used metrics. The goal

is to minimize the degradation of mesh quality after the displacement. Performance is measured on a computational node with two 6-core Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10 GHz processors.

5.1 Two-Step Strategy vs. Standard RBF Interpolation

The quality of the mesh within a double elbow duct, deformed using the programmed software, is compared with the outcome of standard RBF. The displacement is representative of a shape variation that could occur in evolutionary-based shape optimizations. The mesh, having $\sim 2.5 \times 10^6$ nodes, is suitable for viscous flow simulations and is comprised of tetrahedra, pyramids, prisms and hexahedra. Reference and final shapes are illustrated in Figure 9. Table 2 presents quality metrics for the reference and displaced meshes which indicate the ability of the proposed method to achieve results of better quality, for the considered case and metrics, with respect to standard RBF interpolation. This is justified by the fact that, with the two-step strategy, the deformation is interpolated sequentially which has a beneficial effect on the mesh quality preservation [23]. In fact, the maximum cell aspect ratio has increased just by 0.3% using the two-step strategy, against the 3.6% increase by the standard model; the average aspect ratio has increased by 2.4% using either method.

Table 3 and Figure 9 illustrate the nodal error norms of the reference and displaced surface mesh at each step of the two-step procedure. In the first step, the RBF interpolation is carried out using $\sim 10^4$ nodes, Figure 1, as a consequence of the data reduction algorithm. The first step significantly reduces the nodal error norms but the resulting surface mesh does not perfectly fit the new geometry. The second step uses all the $\sim 6 \times 10^4$ surface mesh nodes to perform the RBF interpolation that corrects all boundary nodal displacements. In the second step, the sparsity of the linear system that needs to be solved to perform the RBF interpolation is ~99.8%.

5.2 Scalability Studies on the Mesh Size

Figure 7 illustrates the results of the investigation of the time and memory requirements for performing the displacement of increasingly larger turbine stator and double elbow duct CFD meshes. The turbine stator mesh is displaced from the reference to the second deformed shape of Figure 11, whereas the shape modification of the double elbow duct is that of Figure 9. The time required to morph the meshes is similar in both cases for similar mesh sizes. In both cases, the predictor interpolation and the corrector training are the most expensive phases. The former scales linearly with the mesh size, thanks to the FMM. The latter scales super-linearly due to the increased matrix size. The predictor training time is almost constant since the imposed surface mesh displacements are similar for all mesh sizes. The corrector interpolation phase also scales super-linearly with the mesh size due to the increased number of RBF kernel evaluations, but its contribution to the total computational cost remains minimal, thanks to the integer lattice-based strategy.

5.3 Parametric Study on the Predictor Matrix Size

The DrivAer car geometry is a test case developed by the Technical University of Munich [51] that is herein used in the fastback configuration with mirrors, wheels and a smooth under-body. The mesh with $\sim 4.2 \times 10^6$ nodes, $\sim 3.58 \times 10^5$ of which are surface mesh nodes, consists of various types of elements, with up to 22-face polyhedra. The reference mesh is displaced to the deformed one, as a result of a shape optimization for minimizing drag [52] (not included in the paper). Mesh quality metrics are listed in Table 4. The results exhibit small differences in the maximum aspect ratio which increases by 10% with almost the same maximum skewness. Moreover, no degenerated elements are observed.

Figure 8 reports an investigation of the time required to displace a mesh as a function of the predictor training matrix size. Additionally, the same figure shows the time spent in the predictor against the overall time required to displace the mesh. The results display that in this case an optimal size for the coarsening exists. More importantly, the trend of the overall time required, around the optimal predictor training matrix size, is nearly flat, and non-optimal predictor sizes yield an increase in computational cost below 50%, for this case. In fact, the lowest required time found is ~11 min while the highest is ~15 min. The predictor training matrix size is controlled by the user-defined data reduction parameters.

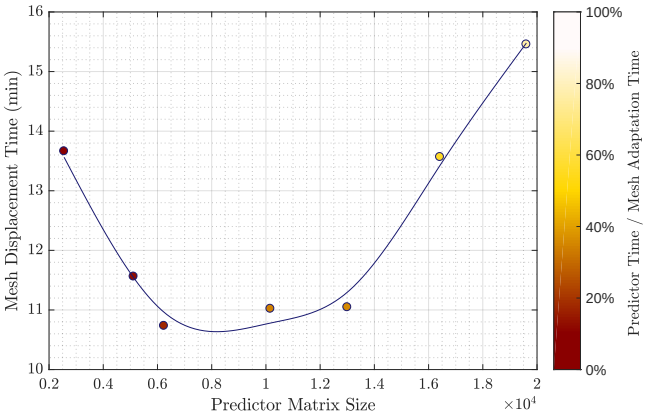

Figure 8: DrivAer car model. Wall-clock time for the mesh displacement, as illustrated in Figure 4, by varying the coarsening in the predictor. Wall-clock time measurements are represented as circles whereas the curve represents the wall-clock time trend. The time spent in the predictor step against the overall mesh displacement time is also marked with colors from red to white.

Table 4: DrivAer car model. Quality metrics for the unstructured polyhedral mesh reported for the reference and deformed designs illustrated in Figure 10.

	Reference	Deformed
Min. Jacobian	>0	>0
Max aspect ratio	36.0	39.4
Max skewness	3.5	3.5

5.4 Aerodynamic CFD Optimization

In this section, the proposed mesh displacement strategy is incorporated into an evolutionary algorithm (EA)-based shape optimization of the turbine stator, using the software EASY (Evolutionary Algorithm System) [53] developed by the researchers' group. The transonic flows, for the various designs, are resolved with the in-house GPU enabled Reynolds-Averaged Navier-Stokes equations solver [54] employing the Spalart-Allmaras turbulence model.

The blade have a maximum chord length of 60 mm. Hub and shroud average radii are 221 mm and 268 mm, respectively. Inlet and outlet conditions are provided in the form of radial profiles, corresponding to a 565 K average inlet total temperature, a 190 kPa average inlet total pressure, 0.1° and 0.4° average inlet peripheral and radial flow angles respectively and a 114 kPa average outlet static pressure. The blade shape is parameterized with NURBS surfaces, and 8 design variables are exposed to the optimization procedure. The surface mesh conforming to each new boundary is obtained by inverting and displacing nodes in the NURBS parametric space, taking special care of trimmed surfaces, as described in [55]. The volume mesh is deformed using the two-step method of this paper with additional treatments for the matching periodic boundaries. The mesh is block-structured with viscous layers and $\sim 2.2 \times 10^6$ nodes, $\sim 1.2 \times 10^5$ out of which are surface nodes.

The optimization aimed at minimizing the mass-averaged total pressure losses and maximizing row capacity. The total pressure losses between the inlet S_I and outlet S_O are expressed in the form of the non-dimensional quantity

$$\Delta P_t = \frac{\overline{p_t}|_{S_I} - \overline{p_t}|_{S_O}}{\overline{(p_t - p)}|_{S_I}}, \tag{13}$$

where p is the pressure, p_t the total pressure, and operators $\overline{\cdot}|_{S_I}$ and $\overline{\cdot}|_{S_O}$ represent mass-flow-averaging at inlet and outlet, respectively. Capacity is defined as

$$C = \dot{m} \left. \frac{\sqrt{\overline{T_T}}}{\overline{p_T}} \right|_{S_O}, \tag{14}$$

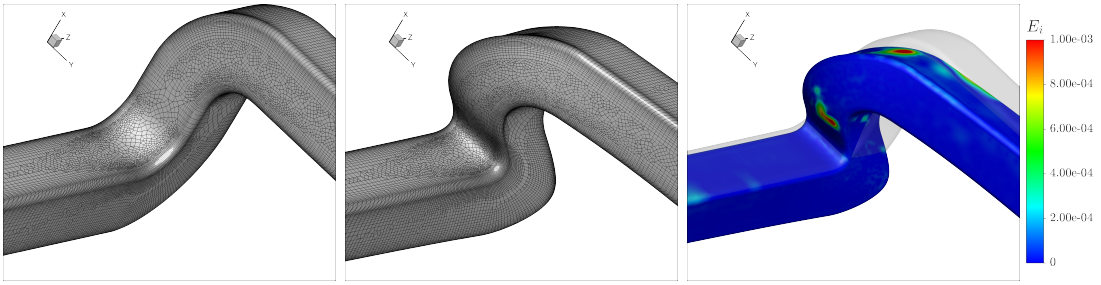

Figure 9: Double elbow duct. Detail of the reference (left) and displaced (middle) CFD unstructured meshes. On the right, the two shapes are compared and the low-to-high nodal error norms of the CFD mesh, after the 1^{st} step of the two-step strategy, is illustrated on the displaced shape with colors ranging from blue to red.

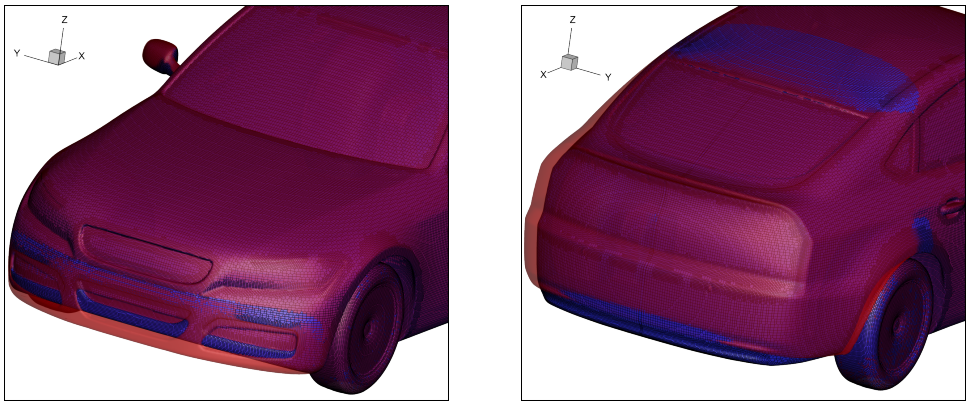

Figure 10: DrivAer car model. Reference (blue) and displaced (red) shape as a result of drag optimization. The most important shape modifications are located at the front (left figure) and rear (right figure) parts of the car model.

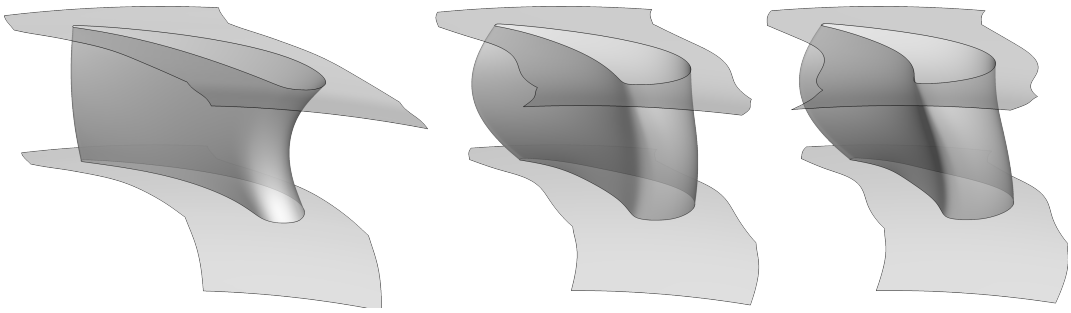

Figure 11: Turbine stator blade. From left to right: reference, 1^{st} and 2^{nd} deformed shapes selected from the front of non-dominated solutions of the multi-objective optimization of the turbine stator (Section 5.4). Mesh quality metrics and optimization results are presented in Table 5.

Table 5: Turbine stator blade. Quality metrics and objective function values for the block-structured volume mesh of the turbine stator reported for the reference and two non-dominated designs illustrated in Figure 11. The sign of the Jacobian is used to check the validity of the mesh. Larger orthogonality metric values and lower normal skewness values are favorable regarding CFD solution accuracy. $y^+ < 1$ of the first nodes off the wall is desirable to guarantee that the mesh near the wall is adequate for CFD simulations with a low-Re turbulence model. In the 2^{nd} deformed mesh, the max. y^+ is higher than 1 only for a small number of nodes, so the results are still considered reliable. Surface mesh quality metrics are also reported since they represent bounds for the quality metrics of the volume mesh.

	Reference	1^{st} Def.	2^{nd} Def.
Capacity $[ms\sqrt{K}]$	1.37×10^{-5}	2.21×10^{-5}	2.58×10^{-5}
Total Pressure Losses	6.82×10^{-2}	4.92×10^{-2}	9.67×10^{-2}
Min. Jacobian	>0	>0	>0
Min. min. orthogonality	0.17	0.14	0.08
Avg. min. orthogonality	0.74	0.68	0.62
Max. max. normal skewness	0.84	0.86	0.93
Avg. max. normal skewness	0.27	0.32	0.39
Max. y^+	0.67	0.95	1.19
Min. surface orthogonality	0.17	0.12	0.07
Max. max. surface normal skewness	0.84	0.89	0.93

where $\dot{m}$ is the mass flow and T_t the total temperature. A geometric constraint to keep the axial chord of the stator blade constant is also imposed. The row is composed of 34 blades.

Figure 11 illustrates the shapes for the reference and three improved designs. To evaluate the quality of the obtained deformed meshes, adapted to the improved shapes, various quality metrics for structured meshes were computed as presented in Table 5 along with the results of the optimization. The results indicated the absence of degenerated elements (minimum Jacobian > 0) even for large deformations. A negative impact on the worse quality metrics (min. min. orthogonality and max. max. normal skewness) resulted, although this is due to just a few elements since averaged quality metrics (avg. min. orthogonality and avg. max. normal skewness) are preserved. Moreover, it should be taken into account that when a volume mesh is deformed, the quality of surface deformations represents a bound for the volume mesh quality. For this reason, in Table 5, the quality metrics for the surface mesh are also tabulated, exhibiting the same trend as the volume mesh quality metrics.

6 Conclusions

This paper places itself among a series of recent efforts to develop faster algorithms for mesh displacement based on RBF. It combines a two-step strategy with an effective preconditioner based on SPAI to accelerate the training phases in both steps. The FMM and an integer lattice are used to speed up the predictor and corrector interpolation phases, respectively.

The predictor reduces the problem size by solving an interpolation problem on a reduced dataset generated by a fast spatial decomposition method based on octree. This results in a geometric approximation of the boundaries which is corrected by the local deformation in the second step. For instance, in the double elbow duct case, the standard RBF requires the solution of a linear system approximately of rank 10^5 whereas, with the proposed method, the training phase is subdivided into the solution of a dense linear system approximately of rank 10^4 and a sparse linear system approximately of rank 10^5, although with just 0.5% of non-zero entries. The SPAI preconditioner reduces the number of iterations needed by the iterative solver by more than one order of magnitude. The computation of the SPAI preconditioner is easily performed in parallel and a strategy, based on geometric considerations, to compute the non-zero structure and reduce the number of dense decompositions needed for its computation, by a factor 15 for the duct case, was proposed.

The performance of the FMM-based interpolation is assessed against standard RBF interpolation, resulting to order of magnitudes saving in computational cost for large CFD meshes. For example, by using the FMM for an RBF model with 1.25×10^5 source and 2×10^6 target nodes, the time needed to compute the displacements of the internal nodes is reduced by 10 times yielding 10^{-7} maximum relative

error or 100 times for 10^{-3}. Using the integer lattice to avoid the computation of the zero-valued RBF kernels is beneficial for both the training and interpolation phase of the corrector step. For instance, for the duct case, it was possible to cut the time needed to assemble the training matrix and perform the RBF interpolation by a factor 100.

The resulting mesh displacement tool operates regardless of the mesh type: it is fast compared to the typical time needed to perform the CFD simulations and preserves mesh quality as well as viscous layers even for large deformations.

The duct and turbine stator cases were used to indicate the scalability of the software for increasing mesh sizes. For example, for a turbine stator, the time needed to displace a grid with $\sim 2.2 \times 10^6$ nodes is less than 2 min while the same displacements applied to a grid with $\sim 2.2 \times 10^7$ nodes requires ~ 12 min. The performance of the software was also examined by varying the coarsening of the predictor step. Results indicate that the performance differs by varying the predictor size. However, for the DrivAer car case, the time required to displace the mesh is nearly constant for a wide range of predictor training matrix sizes, and non-optimal parameters yield an increase in computational time of at most 50%. Multi-objective evolutionary optimization of the turbine stator was performed to indicate the robustness of the software over large displacements. Optimal solutions were found that reduce the total pressure losses up to 27% and increase the capacity up to 88% without the need to generate new meshes even for large shape modifications.

7 Acknowledgments

This research was funded by 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 was an IODA Early Stage Researcher.

The authors wish to express their sincere appreciation to Rolls Royce Deutschland (Dr. M. Meyer and I. Vasilopoulos) for providing the turbine stator test case (Section 5.4) and K. Tsiakas for the help with the in-house GPU-enabled RANS solver.

References

[1] MM Selim and RP Koomullil. "Mesh Deformation Approaches – A Survey". In: *J Phys Math* 7.181 (2016), pp. 2090–0902.

[2] Jianping Niu, Juanmian Lei, and Jiandong He. "Radial basis function mesh deformation based on dynamic control points". In: *Aerospace Science and Technology* 64 (2017), pp. 122–132.

[3] Giorgos A Strofylas, Georgios N Lygidakis, and Ioannis K Nikolos. "An agglomeration strategy for accelerating RBF-based mesh deformation". In: *Advances in Engineering Software* 107 (2017), pp. 13–37.

[4] T. C. S. Rendall and C. B. Allen. "Parallel efficient mesh motion using radial basis functions with application to multi-bladed rotors". In: *International Journal for Numerical Methods in Engineering* 81.1 (2010), pp. 89–105. ISSN: 1097-0207.

[5] CB Allen. "Parallel universal approach to mesh motion and application to rotors in forward flight". In: *International Journal for Numerical Methods in Engineering* 69.10 (2007), pp. 2126–2149.

[6] Matthew L Staten et al. "A comparison of mesh morphing methods for 3D shape optimization". In: *Proceedings of the 20th International Meshing Roundtable*. Springer, 2011, pp. 293–311.

[7] Johnt Batina. "Unsteady Euler airfoil solutions using unstructured dynamic meshes". In: *AIAA Journal* 28.8 (1990), pp. 1381–1388.

[8] Christoph Degand and Charbel Farhat. "A three-dimensional torsional spring analogy method for unstructured dynamic meshes". In: *Computers & Structures* 80.3 (2002), pp. 305–316.

[9] Dehong Zeng and C Ross Ethier. "A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains". In: *Finite Elements in Analysis and Design* 41.11 (2005), pp. 1118–1139.

[10] Carlo L Bottasso, Davide Detomi, and Roberto Serra. "The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes". In: *Computer Methods in Applied Mechanics and Engineering* 194.39 (2005), pp. 4244–4264.

[11] SH Huo et al. "Layered elastic solid method for the generation of unstructured dynamic mesh". In: *Finite Elements in Analysis and Design* 46.10 (2010), pp. 949–955.

[12] Su-Yuen Hsu and Chau-Lyan Chang. "Mesh deformation based on fully stressed design: the method and 2-D examples". In: *International Journal for Numerical Methods in Engineering* 72.5 (2007), pp. 606–629.

[13] Keith Stein, Tayfun E Tezduyar, and Richard Benney. "Automatic mesh update with the solid-extension mesh moving technique". In: *Computer Methods in Applied Mechanics and Engineering* 193.21 (2004), pp. 2019–2032.

[14] Clarence Burg. "Analytic study of 2D and 3D grid motion using modified Laplacian". In: *International Journal for Numerical Methods in Fluids* 52.2 (2006), pp. 163–197.

[15] Brian T Helenbrook. "Mesh deformation using the biharmonic operator". In: *International Journal for Numerical Methods in Engineering* 56.7 (2003), pp. 1007–1021.

[16] Yong Zhao and Ahmed Forhad. "A general method for simulation of fluid flows with moving and compliant boundaries on unstructured grids". In: *Computer Methods in Applied Mechanics and Engineering* 192.39 (2003), pp. 4439–4466.

[17] Xueqiang Liu, Ning Qin, and Hao Xia. "Fast dynamic grid deformation based on Delaunay graph mapping". In: *Journal of Computational Physics* 211.2 (2006), pp. 405–423.

[18] Jeroen AS Witteveen. "Explicit and robust inverse distance weighting mesh deformation for CFD". In: *48th AIAA Aerospace Sciences Meeting*. Vol. 165. 2010, p. 2010.

[19] Joe F Thompson, Zahir UA Warsi, and C Wayne Mastin. *Numerical grid generation: foundations and applications*. Vol. 45. North-Holland Amsterdam, 1985.

[20] HM Tsai et al. "Unsteady flow calculations with a parallel multiblock moving mesh algorithm". In: *AIAA Journal* 39.6 (2001), pp. 1021–1205.

[21] ME Biancolini, IM Viola, and M Riotte. "Sails trim optimisation using CFD and RBF mesh morphing". In: *Computers & Fluids* 93 (2014), pp. 46–60.

[22] Daniel Sieger, Stefan Menzel, and Mario Botsch. "RBF morphing techniques for simulation-based design optimization". In: *Engineering with Computers* 30.2 (2014), pp. 161–174.

[23] A. de Boer, M.S. van der Schoot, and H. Bijl. "Mesh deformation based on Radial Basis Function interpolation". In: *Computers and Structures* 85.1114 (2007). 4th MIT Conference on Computational Fluid and Solid Mechanics, pp. 784 –795. ISSN: 0045-7949.

[24] Michael S. Floater and Armin Iske. "Multistep scattered data interpolation using compactly supported radial basis functions". In: *Journal of Computational and Applied Mathematics* 73.1 (1996), pp. 65 –78. ISSN: 0377-0427.

[25] Francis J Narcowich, Robert Schaback, and Joseph D Ward. "Multilevel interpolation and approximation". In: *Applied and Computational Harmonic Analysis* 7.3 (1999), pp. 243–261.

[26] Damiana Lazzaro and Laura B. Montefusco. "Radial basis functions for the multivariate interpolation of large scattered data sets". In: *Journal of Computational and Applied Mathematics* 140.1 (2002), pp. 521 –536.

[27] M Cordero-Gracia et al. "An interpolation tool for aerodynamic mesh deformation problems based on octree decomposition". In: *Aerospace Science and Technology* 23.1 (2012), pp. 93–107.

[28] Thomas CS Rendall and Christian B Allen. "Reduced surface point selection options for efficient mesh deformation using radial basis functions". In: *Journal of Computational Physics* 229.8 (2010), pp. 2810–2820.

[29] Mario Botsch and Leif Kobbelt. "Real-Time Shape Editing using Radial Basis Functions". In: *Computer Graphics Forum*. Vol. 24. 3. Wiley Online Library. 2005, pp. 611–621.

[30] Laurence Kedward, Christian B Allen, and TCS Rendall. "Efficient and exact mesh deformation using multiscale RBF interpolation". In: *Journal of Computational Physics* 345 (2017), pp. 732 –751.

[31] T Gillebaart et al. “Adaptive radial basis function mesh deformation using data reduction”. In: *Journal of Computational Physics* 321 (2016), pp. 997–1025.

[32] Gregory F. Fasshauer. *Meshfree Approximation Methods with MATLAB*. Illinois Institute of Technology: World Scientific Pub Co Inc, 2007.

[33] M.D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009. ISBN: 9780521101332.

[34] Holger Wendland. “Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree”. In: *Advances in Computational Mathematics* 4.1 (1995), pp. 389–396.

[35] Alexander Kallischko. “Modified Sparse Approximate Inverses (MSPAI) for Parallel Preconditioning”. PhD thesis. Germany: Technische Universitat Munchen, Nov. 2007.

[36] William Fong and Eric Darve. “The black-box fast multipole method”. In: *Journal of Computational Physics* 228.23 (2009), pp. 8712 –8725. ISSN: 0021-9991.

[37] Michele Benzi and Miroslav Tuma. “A comparative study of sparse approximate inverse preconditioners”. In: *Applied Numerical Mathematics* 30.2-3 (1999), pp. 305–340.

[38] Jussi Rahola. “Experiments on iterative methods and the fast multipole method in electromagnetic scattering calculations”. In: *CERFACS, Toulouse, France, Tech. Rep. TR/PA/98/49* (1998).

[39] Michele Benzi. “Preconditioning techniques for large linear systems: A survey”. In: *Journal of Computational Physics* 182.2 (2002), pp. 418–477.

[40] Bruno Carpentieri. “Algebraic preconditioners for the Fast Multipole Method in electromagnetic scattering analysis from large structures: trends and problems”. In: *Electronic Journal of Boundary Elements* 7.1 (2009), pp. 13–49.

[41] Stephen Demko, William F Moss, and Philip W Smith. “Decay rates for inverses of band matrices”. In: *Mathematics of Computation* 43.168 (1984), pp. 491–499.

[42] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012.

[43] Bruno Carpentieri, Iain S Duff, and Luc Giraud. “Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism”. In: *Numerical Linear Algebra with Applications* 7.7-8 (2000), pp. 667–685.

[44] Guillaume Alléon, Michele Benzi, and Luc Giraud. “Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics”. In: *Numerical Algorithms* 16.1 (1997), pp. 1–15.

[45] Bruno Carpentieri et al. “Sparse symmetric preconditioners for dense linear systems in electromagnetism”. In: *Numerical linear algebra with applications* 11.8-9 (2004), pp. 753–771.

[46] Leslie Greengard and Vladimir Rokhlin. “A fast algorithm for particle simulations”. In: *Journal of Computational Physics* 73.2 (1987), pp. 325–348.

[47] Nail A Gumerov and Ramani Duraiswami. “Fast radial basis function interpolation via preconditioned Krylov iteration”. In: *SIAM Journal on Scientific Computing* 29.5 (2007), pp. 1876–1899.

[48] Hongwei Cheng, Leslie Greengard, and Vladimir Rokhlin. “A fast adaptive multipole algorithm in three dimensions”. In: *Journal of Computational Physics* 155.2 (1999), pp. 468–498.

[49] Emmanuel Agullo et al. “Task-based FMM for multicore architectures”. In: *SIAM Journal on Scientific Computing* 36.1 (2014), pp. C66–C93.

[50] Bruno Carpentieri et al. “Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations”. In: *SIAM Journal on Scientific Computing* 27.3 (2005), pp. 774–792.

[51] Angelina I Heft, Thomas Indinger, and Nikolaus A Adams. *Introduction of a new realistic generic car model for aerodynamic investigations*. Tech. rep. SAE Technical Paper, 2012.

[52] EM Papoutsis-Kiachagias et al. “Aerodynamic optimization of car shapes using the continuous adjoint method and an RBF morpher”. In: *11 th International Conference EUROGEN*. 2015, pp. 14–16.

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

[54] Varvara G Asouti et al. “Unsteady CFD computations using vertex-centered finite volumes for unstructured grids on Graphics Processing Units”. In: *International Journal for Numerical Methods in Fluids* 67.2 (2011), pp. 232–246.

[55] Konstantinos T Tsiakas et al. “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*. 2016, pp. 3972–3984.