

Curvilinear Mesh Adaptation using Radial Basis Function Interpolation and Smoothing

Vidhi Zala · Varun Shankar · Shankar P. Sastry ·
Robert M. Kirby

Received: date / Accepted: date

Abstract We present a new iterative technique based on radial basis function (RBF) interpolation and smoothing for the generation and smoothing of curvilinear meshes from straight-sided or other curvilinear meshes. Our technique approximates the coordinate deformation maps in both the interior and boundary of the curvilinear output mesh by using only scattered nodes on the boundary of the input mesh as data sites in an interpolation problem. Our technique produces high-quality meshes in the deformed domain even when the deformation maps are singular due to a new iterative algorithm based on modification of the RBF shape parameter. Due to the use of RBF interpolation, our technique is applicable to both 2D and 3D curvilinear mesh generation without significant modification.

Keywords Curvilinear mesh generation · Radial basis functions · Conformal mapping · Mesh deformation · Mesh adaptation · Mesh quality

PACS 02.60.Ed · 02.60.Jh · 02.60.x

Mathematics Subject Classification (2000) 65(L/N/M)50 · 30E05 · 41A05

1 Introduction

The increasing use of simulations built upon high-order numerical methods in practical engineering problems necessitates the generation of meshes that conform to irregular domain geometries. To maintain the high-order numerical nature of these simulations, the geometric accuracy of the domain must also be high-order, thus motivating (high-order) curvilinear meshes. The starting point for many high-order meshing techniques is to create a valid low-order (straight-sided) mesh which is then “adapted” to the curved geometry. The challenge when accomplishing this update is the balancing act between faithfully representing the boundaries of interest while maintaining a mesh whose elements are of good quality (and hence have favorable numerical properties). We present a technique based on RBF-interpolation that produces superior quality meshes by first deforming the domain to meet the geometric constraints of the problem and then iteratively adapting or smoothing the mesh in a way to capitalize on the properties of the RBF-interpolation functions

Vidhi Zala
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT E-mail: vidhi@sci.utah.edu

Varun Shankar
Department of Mathematics, University of Utah, Salt Lake City, UT
E-mail: vshankar@math.utah.edu

Shankar P. Sastry
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT

Robert M. Kirby
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT
E-mail: kirby@sci.utah.edu

we employ. In this work, mesh adaptation has been studied in the context of both refinement (coarsening) and smoothing; the technique presented herein aims to combine the best of both approaches.

We begin by summarizing the state-of-the-art in high-order (curvilinear) mesh generation. Over the last decade, many techniques for the generation and deformation of linear meshes into curvilinear meshes have been proposed [27, 28, 40, 41, 54]. Sastry et al. [48] provided the following taxonomy for partitioning the literature landscape: optimization-based methods [46, 47], PDE-based methods [41], and interpolation-based techniques [4]. Some of the notable work done in the mesh deformation and curvilinear mesh generation can be attributed to the application of one or more techniques from these three classes. The optimization-based techniques aim at optimizing an objective function depending on the geometry of the domain and the mesh. Sastry et al. [47] proposed a log-barrier optimization routine to dictate vertex movement and to improve the quality of a tangled mesh (due to the deformation) to obtain a valid mesh. The Remacle group [30, 46] developed a log-barrier technique that generates a valid mesh by maximizing the minimum Jacobian of high-order elements in the mesh. From the class of PDE-based methods, Moxey et al. [39] presented a technique based on the thermo-elastic analogy by modelling the mesh as non linear elastic material. In a subsequent paper, Moxey et al. [56] take the variational approach further by optimizing mesh quality using a scaled Jacobian approach. Sastry et al. [48] compared and contrasted the thermo-elastic method with the RBF interpolation using thin-plate-splines. Experiments for that effort help establish the superiority of the RBF interpolation-based technique by generating elements that are of higher quality and conform to the boundary geometry.

The last class, interpolation-based methods, has been mostly applied to linear mesh deformation functions. Staten et al. [53] developed the simplex-linear transformation algorithm, which carries out a linear interpolation of mesh vertices after making a coarse mesh as initial step. Sastry et al. [48] developed a technique for curvilinear mesh generation using thin-plate spline RBFs, which belong to the class of polyharmonic splines. They further demonstrated that interpolants based on polyharmonic spline help preserve the shape of elements after deformation. However, the thin-plate spline technique did not possess the ability to deal with degenerate deformation maps, or smooth any resulting mesh tangles. Further, the technique did not generalize to 3D meshes in a straightforward fashion.

Broadly speaking, we can treat PDE-based methods and interpolation-based methods as being in the same class, where the positions of the interior mesh vertices are interpolated from the positions of the boundary vertices using either the solution of a PDE or an explicit interpolation technique. Such a characterization is useful as it helps motivate our work: we seek to develop an interpolation-based method that through our choice of the interpolating functions mimics some of the favorable properties observed in the PDE-based approaches while being applicable to both 2D and 3D mesh generation. We present a generalization of [48] that uses RBFs with a shape parameter to smooth node clusters resulting from singular or non-smooth deformation maps. Specifically, we turn to the Matérn kernels (also referred to as *Sobolov splines*), a family of RBFs closely related to the polyharmonic splines. As their alternate name implies, interpolants based on these kernels are the minimum Sobolev norm interpolants, possessing similar properties to polyharmonic splines, but possessing a shape parameter that is extremely useful for tuning. In Section 3, we compare the Matérn kernels to the polyharmonic splines, and present a tuning algorithm for the shape parameter to help achieve quasi-local smoothing of these interpolants.

The remainder of the paper is organized as follows. In Section 2 we review RBF interpolation with a focus on Matérn kernels, our basis of choice; we justify the use of this basis, and we also present a generalization of existing techniques to smooth RBF interpolants. We go on to present a mathematical description of our quality heuristics and a new adaptation and smoothing algorithm in Section 3. We then undertake a thorough complexity analysis of our method in Section 4. Finally, we present numerical experiments exploring the behavior of our method on different classes of deformation functions in Section 5. We conclude with a discussion of the results and provide some comments on future work.

2 Review

2.1 RBF Interpolation

RBFs are a popular tool for scattered data interpolation in arbitrary dimensions and have become increasingly popular in machine learning [35, 49], computer graphics [6, 34], mesh generation and repair [36, 48] and in the numerical solution of PDEs [32, 52]. More relevant to this article, RBFs have also been used to interpolate data on co-dimension one submanifolds of \mathbb{R}^s with excellent approximation properties using only straight-line (i.e. Euclidean) distances in the embedding space [25], a feature that has been leveraged to solve PDEs on surfaces [26, 51]. In our application, the relevant submanifolds are the boundaries of (irregular) domains in \mathbb{R}^2 and \mathbb{R}^3 .

We now briefly describe RBF interpolation in \mathbb{R}^s ; for interpolation on submanifolds $\mathbb{M} \subset \mathbb{R}^s$, it is only necessary for the points to lie on \mathbb{M} . Given a set of (scattered) nodes $X = \{\mathbf{x}_i\}_{i=1}^N$ in \mathbb{R}^s and a set of data values $Y = \{\mathbf{y}_i\}_{i=1}^N$ sampled from some function $f : \mathbb{R}^s \rightarrow \mathbb{R}$, the RBF approximation to f is obtained by a linear combination of *shifts* of a single *radial* kernel or basis function ϕ such that

$$I_\phi f(\mathbf{x}, \epsilon) = \sum_{i=1}^N \lambda_i(\epsilon) \phi(\epsilon, r_i(\mathbf{x})) \quad (1)$$

where $\phi(\epsilon, r_i(\mathbf{x})) = \phi(\epsilon \|\mathbf{x} - \mathbf{x}_i\|)$ and $\epsilon > 0$ is a *shape parameter* that controls the flatness of the RBF. To find the unknown coefficients λ_i , we enforce the interpolation conditions

$$I_\phi f|_X = Y, \quad (2)$$

$$\implies I_\phi f(\mathbf{x}_i, \epsilon) = \{\mathbf{y}_i\}_{i=1}^N. \quad (3)$$

If ϕ is a positive-definite radial kernel or an order one conditionally positive-definite kernel on \mathbb{R}^s and all nodes in X are distinct, the above interpolation problem has a unique solution, and the corresponding RBF interpolation matrix is invertible [12]. In the limit as $\epsilon \rightarrow 0$ (i.e. a flat kernel), RBF interpolants to data scattered in \mathbb{R}^s typically converge to (multivariate) polynomial interpolants [11, 33, 50], and to spherical harmonic interpolants on a sphere [21]. For smooth target functions, smaller (but non-zero) values of ϵ generally lead to more accurate RBF interpolants [22, 33]. Unfortunately, computing these interpolants by solving the linear system involving the RBF interpolation matrix becomes ill-conditioned for small ϵ (see, e.g., [23]). While some stable algorithms have been developed for bypassing this ill-conditioning [14, 18–22], these algorithms do not apply when the nodes lie on a lower-dimensional surface than the embedding space. Our approach will be to pick a value of ϵ that results in some target condition number κ in the interpolation matrix that is very close to the edge of ill-conditioning. This typically results in excellent approximation [51]. Our goal will be to approximate vector-valued functions in this work. We accomplish this by interpolating each component of the vector-valued functions using a scalar RBF interpolant.

For similar reasons to [48], we choose an RBF ϕ with global support. Specifically, we use the piecewise-smooth C^4 Matérn kernel given by:

$$\phi(\epsilon r) = (3 + 3\epsilon r + \epsilon^2 r^2) e^{-\epsilon r}. \quad (4)$$

Our reasons for using this kernel are twofold: first, the reproducing kernel Hilbert space corresponding to this kernel is a standard Sobolev space and therefore well-understood; second, unlike the polyharmonic splines, the Matérn kernel comes equipped with a shape parameter ϵ , such that the limit $\epsilon \rightarrow 0$ recovers the polyharmonic spline kernels used in [48]. Modification of this shape parameter upon evaluation of the RBF interpolant can allow *smoothing*. This will be explained in the following section. For more on Matérn kernels, we refer the reader to [12, 13].

2.2 Mesh Quality

Our RBF-based technique accomplishes two distinct purposes: first, it recovers deformation maps (and therefore a deformed mesh) using data only on the boundary of the input and output domains; second, it

also attempts to automatically smooth the recovered deformation map so as to obtain a deformed mesh with good-quality elements. An element quality metric is a scalar function of node positions that measures some geometric property of the element [31]. In this section, we present a brief overview of the popular metrics for measuring mesh quality. Assume for the following discussion that a mesh contains a finite set of vertices V defined as $V = \{\mathbf{x}_i\}_{i=1}^N$ in \mathbb{R}^s , and a finite set of elements E defined by groupings of those vertices. The elements are triangles in 2D and tetrahedra in 3D.

There are many popular techniques for generating meshes out of point sets, like octree mesh generation [9], Delaunay triangulation [8, 16, 29] and advancing-front [38]. Out of these techniques, the Delaunay triangulation is most commonly used as it provides triangulations whose elements respect certain quality criteria. Given a set of points V , the Delaunay technique attempts to create triangulations wherein each triangle maximizes one (or more) of the following ratios: the inradius to the circumradius; the shortest edge to the longest edge; the shortest altitude to the longest edge; the aspect ratio, etc. [1–3, 9, 10, 15, 17, 24, 37, 42–44, 57]. In this article, we use the inradius to circumradius ratio as our element-wise quality metric, given by:

$$Q = \frac{8A^2s}{abc(a+b+c)}, \quad (5)$$

where a, b, c are side lengths, s is the dimension and A is the area of the element. The use of the inradius to circumradius ratio for measuring the quality of elements was suggested by Cavendish, Field and Frey [5]. A high value of $Q \in [0, 1]$ implies better quality elements. An equilateral triangle and a standard tetrahedron has $Q = 1$. They are considered the standard elements for 2D and 3D meshes respectively.

3 Methods

3.1 Smoothing with the Shape Parameter

Our goal is to develop an iterative quasi-local smoothing algorithm to rectify singular deformation maps. To do so, we utilize an interesting feature of RBF interpolation: smoothing using the shape parameter. This was first proposed by Beatson in the context of surface reconstruction from point cloud data [7], and has since been used as part of a numerical method for solving coupled PDEs [52]. This technique is very simple to apply: first, find the interpolation coefficients $\lambda_i(\epsilon^*)$, where ϵ^* is some small non-zero value. Then, when evaluating the interpolant, replace ϵ^* with ϵ , where $\epsilon \neq \epsilon^*$. In other words, given an evaluation node set $X = \{\mathbf{x}_j\}_{j=1}^M$, evaluate the interpolant at each point \mathbf{x}_j as

$$I_\phi f(\mathbf{x}_j, \epsilon) = \sum_{i=1}^N \lambda_i(\epsilon^*) \phi(\epsilon, r_i(\mathbf{x}_j)), \quad (6)$$

where $r_i(\mathbf{x}_j) = \|\mathbf{x}_j - \mathbf{x}_i\|$. If $\epsilon < \epsilon^*$, this amounts to evaluating the coefficients against a slightly smoother basis than the one we interpolated with; this results in a smoothing; conversely, choosing $\epsilon > \epsilon^*$ can result in a sharpening of low-frequency details.

In this article, we present and utilize a simple generalization of the above approach: we allow ϵ to vary from point to point. In other words, we now evaluate the interpolant pointwise as

$$I_\phi f(\mathbf{x}_j, \epsilon_j) = \sum_{i=1}^N \lambda_i(\epsilon^*) \phi(\epsilon_j, r_i(\mathbf{x}_j)), \quad (7)$$

where $\epsilon_j > 0, j = 1, \dots, M$ are now *pointwise* shape parameters that potentially differ from the interpolation shape parameter ϵ^* . Since ϕ has global support, this is still not entirely a local smoothing. However, compared to previous approaches which use a single ϵ , our new approach constitutes a *quasi-local* smoothing of the interpolant. In Section 3.2, we describe a technique which generates each ϵ_j given ϵ^* , X and $Y^d \subset Y^b$ samples of the deformation function on the boundary. Here, $Y^d = pY^b$, $0 < p \leq 1$ and Y^b is set of points on the boundary of deformed domain. The points on boundary are chosen based on the equation that describes the boundary and a boundary thickness parameter α set in step 3 of the Algorithm 1. For example, if the 2D domain is a unit circle centered at origin, all the points that satisfy the equation $x^2 + y^2 = 1$ within a tolerance

of α falls on the boundary. The parameter p is chosen randomly and the subset is formed uniformly. The idea here is to show the efficacy of deformation map in deforming the entire domain even when we pick fewer points on the boundary. As we will see in Section 5, the scalar-valued RBF approximation and smoothing method described here, when applied in component-wise fashion to 2D and 3D problems, gives intuitive results in the form of an appropriately smoothed set of output nodes Y .

3.2 RBF-interpolation Based Iterative Algorithm for Mesh Generation and Quality Improvement

In this section we present the RBF-interpolation based algorithm for generating curvilinear mesh and iterative smoothing, discuss implementation details of the same and provide a detailed analysis in terms of complexity.

Overview

Algorithm 1 describes the algorithm for obtaining a high-quality deformed mesh given a set of points in an initial undeformed domain and a set of parameters that control the deformation and smoothing process. Broadly, the procedure can be seen as a collection of following tasks:

1. Given the undeformed domain and samples of a deformation function on the boundary, interpolate the function (given by Equation (1)) to recover the deformation map in the interior of the domain.
2. Tessellate the deformed domain and calculate element quality $Q = q_e$ (see Section 3.2.1).
3. Distribute the quality metric from the elements to vertices by averaging the quality of elements in 2-ring neighborhood around each vertex (see Section 3.2.1).
4. For vertices with quality (q_v) less than a predefined tolerance, reduce the shape parameter (ϵ) by some factor (see Section 3.2.2).
5. Evaluate the interpolant using the list of modified shape parameters (ϵ_j) described by Equation (7) to obtain an improved deformed mesh.
6. Repeat Steps 2 through 5 until convergence defined by stopping criteria.

3.2.1 Computing quality per-element and per-vertex

At each iteration of our RBF-based technique, the resulting mesh element quality (q_e) is determined based on one of the definition of quality metric as detailed in Section 2.2. The overall quality of the mesh is the aggregate of quality of all elements in the mesh. This is used to determine the stopping criteria for the algorithm. If the overall quality satisfies a predefined threshold, the algorithm converges.

Let \mathbf{y}_k be the 2-ring neighbors of a node \mathbf{y} in Y . We view these nodes as constituents of a stencil for measuring the per-vertex quality. Here, the number of vertices in the stencil (n_k) depends on the degree of connectedness of the vertex. For instance, vertices which are close to the domain boundary have fewer neighbors while others have a full connectivity with 2-ring neighbors. Because the quality is defined per-element and we want to have a quality measure per-vertex, we need to find the elements connected by a vertex. To aggregate elemental qualities q_e to individual vertex qualities q_y for vertices in Y , we use the following average:

$$q(\mathbf{y}_k) = (q_y)_k = \frac{1}{n_k} \sum_{i=1}^{n_k} q_e(\mathbf{y}_i), k = 1, \dots, |Y|, \quad (8)$$

where $|Y|$ is the total number of vertices in the domain.

3.2.2 Modifying shape parameter based on the per-vertex quality

We now describe our formula for generating a new modified shape parameter at each vertex. We modify the shape parameter at a vertex based on two factors: the quality measure at the vertex (given by Equation (8)), and the proximity of the vertex to the boundary. Without loss of generality, we focus on the vertex \mathbf{y}_k . Let ϵ_k^{old} be the n_k -long vector of shape parameters of \mathbf{y}_k and its 2-ring neighbors in the current iteration. At the first iteration of the smoothing algorithm, the shape parameters at all vertices are the same, *i.e.*,

Algorithm 1 RBF-based iterative algorithm for mesh generation and quality improvement

```

1: Set  $\delta \leftarrow$  scaling factor for shape parameter update term
2: Set  $\sigma \leftarrow$  falloff of local Gaussian smoothing for shape parameter
3: Set  $\alpha \leftarrow$  thickness of boundary
4:  $X_i \leftarrow N_i \times s$  matrix of interior nodes on  $\Omega(\mathbb{R}^s)$ 
5:  $X_b \leftarrow N_b \times s$  matrix of boundary nodes on  $\partial\Omega(\mathbb{R}^s)$ 
6:  $X = X_i \cup X_b \leftarrow N \times s$  matrix containing all nodes,  $N = N_i + N_b \leftarrow |\Omega|$ 
7:  $X^d \subset X_b$ ,  $N_d \times s$  matrix of data sites on  $\partial\Omega$ ,  $N_d \subset N_b = |\partial\Omega|$ 
8:  $Y^d \leftarrow N_d \times s$  matrix of deformed boundary nodes (corresponding to  $X^d$ )
9:  $\kappa_t \leftarrow$  desired target condition number of interpolated matrix
10:  $\epsilon^* \leftarrow$  ideal shape parameter corresponding to  $\kappa_t$ 
11: Initialize  $\underline{\epsilon} = \epsilon^* \forall \mathbf{x}$  in  $X$ 
12:  $A \leftarrow$  RBF interpolation matrix using  $\epsilon^*$ 
13:  $\underline{\lambda} \leftarrow$  Interpolation coefficients, obtained formally by finding  $A^{-1}X^d$  once
14:  $Y \leftarrow$  Evaluate the RBF interpolant built on  $X^d$  at all nodes in  $X$  with  $\underline{\epsilon}$ 
15: Tessellate  $Y$  to obtain element set  $E$ 
16: For each  $\mathbf{y}$  in  $Y$ , store its  $n_k$  2-ring neighbors  $\leftarrow \{\mathbf{y}, \mathbf{y}_k\}$ 
17: for each correction iteration until convergence do
18:   Calculate  $\underline{q}_e \leftarrow$  quality per element in  $E$ 
19:   Append  $\|\underline{q}_e\|_2$  to  $\underline{h}$ , history of mesh quality over iterations
20:   Check convergence:  $\|\underline{q}_e\|_2 < \max(\underline{h})$ 
21:   Distribute  $\underline{q}_e$  to  $\underline{q}_y \leftarrow$  quality per vertex in  $Y$ 
22:   for each point  $\mathbf{y}_k$  in  $Y$  do
23:      $\mu(\mathbf{y}_k, \alpha) \leftarrow \min \|\mathbf{y}_k - \mathbf{y}_p\|_2 \forall \mathbf{y}_p \in Y^d$ 
24:      $\mu(\mathbf{y}_k, \alpha) = 0$  if  $\mu(\mathbf{y}_k, \alpha) \leq \alpha$ 
25:     for each  $j$  from 1 to  $n_k$  do
26:        $(\underline{\Psi}_k)_j \leftarrow |q_k - q_j|$ 
27:     end for
28:      $\underline{\gamma}_k \leftarrow e^{-\sigma \underline{\Psi}_k}$ 
29:      $\theta_k \leftarrow \delta \mu(\mathbf{y}_k, \alpha)$ 
30:      $\underline{\epsilon}_k \leftarrow \epsilon_k - \theta_k \underline{\gamma}_k$ 
31:   end for
32:   Compute smoothed node set  $Y$  using new  $\underline{\epsilon}$  and precalculated  $\underline{\lambda}$ 
33:   Tessellate  $Y$  to obtain element set  $E$ 
34:   For each  $\mathbf{y}$  in  $Y$ , update its  $n_k$  2-ring neighbor stencil  $\leftarrow \{\mathbf{y}, \mathbf{y}_k\}$ 
35: end for

```

$\underline{\epsilon}_k^{old} = \epsilon^*$, $k = 1, \dots, N$. The goal is to obtain $\underline{\epsilon}_k^{new}$, the new vector of shape parameters, for every subsequent iteration. We propose a simple update of the form

$$\underline{\epsilon}_k^{new} = \underline{\epsilon}_k^{old} - \theta_k \underline{\gamma}_k, \quad (9)$$

where θ_k is a factor that accounts for proximity to boundaries, and $\underline{\gamma}_k$ is a factor that depends on the vertex qualities of \mathbf{y}_k and its 2-ring neighbors; this formula is given on line 30 of Algorithm 1. We will first explain the $\underline{\gamma}_k$ term, then the θ_k term.

The term $\underline{\gamma}_k$ is a function of the vertex quality $(q_y)_k$ associated with the vertex \mathbf{y}_k . Specifically, this term attempts to decrease $\underline{\epsilon}_k^{old}$ whenever the vertex quality associated with \mathbf{y}_k is significantly different from the vertex qualities of its 2-ring neighbors. First, we define the quantity $\underline{\Psi}_k$ as

$$(\underline{\Psi}_k)_j = |(q_y)_k - q_j|, j = 1, \dots, n_k, \quad (10)$$

where j indexes the 2-ring neighbors of the vertex \mathbf{y}_k . Clearly, $\underline{\Psi}_k$ is a vector of differences in quality between \mathbf{y}_k and its 2-ring neighbors. Our formula for $\underline{\gamma}_k$ satisfies two requirements: first, that $\underline{\gamma}_k$ change *smoothly* as a function of $\underline{\Psi}_k$, and second, that $\underline{\gamma}_k$ is smaller as we go further away from \mathbf{y}_k . These two requirements are satisfied by requiring $\underline{\gamma}_k$ to take the form

$$\underline{\gamma}_k = e^{-\sigma \underline{\Psi}_k}, \quad (11)$$

where σ is some user-supplied *falloff* factor. If σ is small relative to the distance between nodes, the different $\underline{\Psi}_k$ values contribute more equally to $\underline{\gamma}_k$. In contrast, if σ is large, the contributions of $\underline{\Psi}_k$ corresponding to nodes other than \mathbf{y}_k are smaller. In this article, we use values of σ that ensure that we are in the latter

regime. This allows us to more effectively correct localized irregularities in vertex quality, while still smoothly updating the $\underline{\epsilon}_k^{old}$ values.

When we attempted to update $\underline{\epsilon}_k^{old}$ using only the $\underline{\gamma}_k$ values, we ran into two difficulties. First, we observed that nodes from the interior would leave the domain boundary, and hence would need to be periodically deleted from the domain. Second, such updates tended to undo mesh refinement near the domain boundary. Our first attempt at fixing this problem was to multiply $\underline{\gamma}_k$ by a *switch* that *turns off* smoothing near the boundary. However, noticing that this produced some mesh tangling outside the boundary-refined layers, we choose instead to multiply $\underline{\gamma}_k$ by the scalar term θ_k (line 29 in Algorithm 1) defined as

$$\theta_k = \delta \mu(\mathbf{y}_k, \alpha). \quad (12)$$

Here, δ is some small number that controls the magnitude of θ_k , and $\mu(\mathbf{y}_k, \alpha)$ is a function that effectively specifies a “boundary-layer” for our algorithm. Let \mathbf{y}_p be the closest boundary point to \mathbf{y}_k . Then, $\mu(\mathbf{y}_k, \alpha)$ (lines 23 and 24 of Algorithm 1) is defined as

$$\mu(\mathbf{y}_k, \alpha) = \begin{cases} \|\mathbf{y}_k - \mathbf{y}_p\|, & \|\mathbf{y}_k - \mathbf{y}_p\| > \alpha \\ 0, & \|\mathbf{y}_k - \mathbf{y}_p\| \leq \alpha \end{cases}$$

This function ensures that no update is made to the shape parameter of any node \mathbf{y}_k within distance of α from its closest point \mathbf{y}_p on the boundary. Further, nodes \mathbf{y}_k further away from their closest boundary points are allowed to receive larger updates to their shape parameter vectors $\underline{\epsilon}_k$.

In general, we find that δ needs to be small to improve quality, ensuring that the shape parameters are not decreased too much in any iteration. Currently, δ , α and σ are selected by trial and error, but one could imagine using training techniques from the neural networks literature to accomplish this. We leave such extensions for future work.

3.2.3 Stopping criterion

We now present a stopping criterion for the smoothing algorithm. The criterion is designed to stop the iterative smoothing if the mesh quality begins to worsen as a consequence of the iterative procedure. Such a worsening in quality, when it occurs, is a consequence of the global support of the RBF interpolant. Despite the local nature of the shape parameter updates, the global support of the RBFs means that most nodes are moved to some extent.

To determine a good stopping point for the iterative smoothing process, we simply check the 2-norm of the per-vertex quality measure, *i.e.*, $\|\underline{q}_e\|_2$. If $\|\underline{q}_e\|_2$ is smaller for the current iteration than for previous ones, the algorithm halts. This choice of stopping criterion may not be ideal, since it aggregates mesh quality into a single number. However, we have found that it works well in conjunction with the global RBF interpolant. We leave the question of stopping criteria for future work.

4 Complexity Analysis

4.1 Preprocessing

We first consider the preprocessing costs of our algorithm. Consider a tessellated domain $\Omega \subset \mathbb{R}^s$. Let N_b be the number of points on the boundary of the domain ($\partial\Omega$) and N_i be the points in the interior. The total number of points in the domain is then given by $N = N_i + N_b$. However, not all N_b points are used to recover the deformation map via the RBF interpolant. Let $N_d \subset N_b$ be the number of points used to build the RBF interpolant. The initial preprocessing step involves computing and decomposing the RBF interpolation matrix once for a cost of $O(N_d^3)$. The interpolant can then be evaluated for $O(NN_d)$. However, it is more intuitive to express this cost in terms of the number of interior points. We now present that derivation, specialized to $s = 2, 3$.

4.1.1 Complexity analysis in 2D ($s = 2$)

Before proceeding, assume that points on the boundary are evenly-spaced with spacing h_b . Further, assume that the interior nodes are spacing h_i . Then, we have

$$h_b = \frac{l}{N_b}, h_i = \left(\frac{A}{N_i} \right)^{\frac{1}{2}}, \quad (13)$$

where l is the perimeter of the boundary and A is the area of the domain. Assuming without loss of generality that $h_i = h_b$, we have

$$\frac{l}{N_b} = \left(\frac{A}{N_i} \right)^{\frac{1}{2}} \implies N_b = lA^{-\frac{1}{2}}N_i^{\frac{1}{2}}. \quad (14)$$

Now, letting $N_d = pN_b$, $0 < p \leq 1$, we can rewrite Equation (14) as

$$N_d = pN_b = plA^{-\frac{1}{2}}N_i^{\frac{1}{2}}. \quad (15)$$

Since the interpolation matrix is inverted for a one-time cost of $O(N_d^3)$, we now have an explicit expression for that cost. Using Equation (15), this cost becomes:

$$N_d^3 = p^3 l^3 A^{-\frac{3}{2}} N_i^{\frac{3}{2}} \implies N_d^3 = p^3 A^{\frac{1}{2}} \left(\frac{l}{A} \right)^3 N_i^{\frac{3}{2}}. \quad (16)$$

Let ψ_s be a domain-dependent constant in s dimensions so that for $s = 2$, $\psi_2 = \frac{l}{A}$. We can use this to rewrite Equation (16), obtaining:

$$N_d^3 = p^3 A^{\frac{1}{2}} \psi_2^3 N_i^{\frac{3}{2}}. \quad (17)$$

In general, $N_b \ll N_i$, implying that $N \approx N_i$. Thus, the preprocessing cost C_2 for our technique in $s = 2$ spatial dimensions is asymptotically $C_2 = O(N^{1.5})$. Note that if the interpolation problem could be solved in $O(N_d)$ operations, this cost reduces to $O(\sqrt{N})$. We leave this extension for future work.

4.1.2 Complexity analysis in 3D ($s = 3$)

We now derive 3D complexity estimates for our preprocessing step. Assuming that nodes on the domain boundary (now a surface of co-dimension one in \mathbb{R}^3) are quasi-uniform with spacing h_b and assuming the interior node spacing is h_i , we have

$$h_b = \left(\frac{A}{N_b} \right)^{\frac{1}{2}} \text{ and } h_i = \left(\frac{V}{N_i} \right)^{\frac{1}{3}}, \quad (18)$$

where A is now the surface area of $\partial\Omega$, and V is the volume of Ω . Assuming again that $h_i = h_b$, we have

$$\left(\frac{A}{N_b} \right)^{\frac{1}{2}} = \left(\frac{V}{N_i} \right)^{\frac{1}{3}} \implies N_b = AV^{-\frac{2}{3}}N_i^{\frac{2}{3}}. \quad (19)$$

Expressing this in terms of N_d , the number of data sites used to build the interpolant, we have

$$N_d = pN_b = pAV^{-\frac{2}{3}}N_i^{\frac{2}{3}}. \quad (20)$$

The preprocessing cost is $O(N_d^3)$, which is now given by:

$$N_d^3 = p^3 A^3 V^{-2} N_i^2. \quad (21)$$

Now letting $\psi_3 = \frac{A}{V}$, we have

$$N_d^3 = p^3 V \psi_3^3 N_i^2. \quad (22)$$

The preprocessing cost C_3 can now be expressed in terms of the total number of points N as $C_3 = O(N^2)$. Again, as in 2D, it is possible to lower this cost (to $O(N^{\frac{2}{3}})$) if the interpolation problem is solved in $O(N_d)$ operations.

4.1.3 Finding the initial shape parameter

Another contribution to the preprocessing cost comes from the calculation of the initial shape parameter (ϵ^*). We use the `fzero` function in Matlab to find this shape parameter. This function uses an iterative method called the Brent-Dekker method to find the zero of a function in a given interval. Consequently, it requires an evaluation of that function multiple times. In our application, the function that must be evaluated is the condition number of the RBF interpolation matrix. This can be computed for a cost of $O(N_d^3)$ if the 2-norm condition number is used, and a cost of $O(N_d^2)$ if the 1-norm or max-norm condition numbers are used. From Equation (15), and considering max-norm, it is obvious that this cost scales as $O(N)$ in 2D; similarly, Equation (20) for max-norm condition number shows that this cost scales as $O(N^{\frac{4}{3}})$ in 3D.

4.2 Complexity of the Smoothing Algorithm

We now analyze the complexity of a single step of our smoothing algorithm. To do so, we break down our algorithm into several key steps.

4.2.1 Finding the 2-ring neighbors

Each iteration of the algorithm requires finding the 2-ring neighbors of each vertex using the Delaunay triangulation (which is itself constantly being updated). To find the 2-ring neighbors of each vertex, we first find the list of vertices connected to each vertex (the 1-ring neighbors). The cost of this operation for N vertices scales as $O(N \log N)$, with a dimension dependent constant. The next step is to find the set of immediate neighbors of the 1-ring neighbors. To do so, we simply repeat the above step for each of the 1-ring neighbors. The total asymptotic cost of finding the 2-ring neighbors is therefore:

$$C_{n_k\text{-neighbors}} = N(1 + n_k) \log N. \quad (23)$$

4.2.2 Calculating per-element and per-vertex quality

At each iteration, we calculate per-element quality (q_e) and distribute it to the constituent nodes forming the elements as (q_y) given by Equation (8). q_e can be computed for a cost of $O(N)$. Similarly, q_y requires an averaging over elements connected to each vertex. If the average number of elements connected to each vertex is n , then this cost scales as $O(nN)$, where $n \ll N$. The complexity of this step therefore scales as $O(N)$.

4.2.3 Updating the evaluation shape parameter

At each iteration, the algorithm updates the shape parameter for each vertex in the node set based on the quality metric and calculations described by Equation (9). This operation utilizes the 2-ring neighbor information from previous step and the predefined parameters described in Section 3.2.2. By a similar argument to the previous subsection, this update also scales as $O(nN) \approx O(N)$.

4.2.4 Computing the smoothed node set

To obtain the smoothed node set Y at each iteration, we need to compute the RBF evaluation matrix and multiply it with the precomputed interpolation coefficients. The computation of the evaluation matrix is straightforward, as shown in line 32 of Algorithm 1. The operation scales as $O(NN_d)$ when N evaluation points are used. It is clear from Equation (15) that this cost scales as $O(N^{\frac{3}{2}})$ in 2D with a small constant term. In 3D, Equation (20) shows that this cost scales as $O(N^{\frac{5}{3}})$, again with a small constant.

4.2.5 Tessellation of domain to obtain element set

As a last step of each smoothing iteration, the node set Y is tessellated to obtain a mesh which is smoother than the previous iteration. This operation is performed using the Delaunay triangulation in the code (refer to line 33 in Algorithm 1). There are many other algorithms to obtain a mesh from the node set and depending on the use case, this choice can vary. In general, this cost is $O(N \log N)$ in 2D, and $O(N^2)$ in 3D.

5 Results

We now present the results of our numerical experiments using our algorithm. To demonstrate the efficacy of the technique across different types of problems, we focus on three test cases involving domains with boundaries of different smoothness:

1. Deforming a C^1 boundary to a C^∞ boundary.
2. Deforming a domain with a C^∞ outer boundary and a C^∞ inner boundary (an annulus) to a C^1 outer boundary and C^1 inner boundary (a square with an airfoil cavity).
3. Deforming a cube (C^1 boundary) to a sphere (C^∞ boundary).

A fourth category for which results are not shown is the deformation of domains with C^∞ boundaries to domains with C^∞ boundaries. We do not show results for this case, as the smoothing procedure is completely unnecessary here. The interpolation step itself produces excellent meshes, at least partly due to the spectral convergence rates achieved by the RBF interpolant to the deformation map.

In order to obtain tessellations on the undeformed domains, we first generate a set of reasonably well-distributed nodes using a repulsion algorithm such as the one used by Distmesh [45]. This gives us both interior and boundary nodes. The undeformed mesh is then generated by applying a simple Delaunay triangulation on this set of nodes. Following Section 3.1, we computed initial shape parameters for all the above test cases. The results are summarized in Table 1.

Table 1: Summary of test parameters. C^4 Matérn kernel was used for all the experiments

Test	ϵ^*	$N = N_i + N_b$	N_d	Mesh norm	α	δ	σ
C^1 boundary to a C^∞ boundary	0.2497	2106 = 1906+200	172	1.0058	0.001	9.9422e-07	0.1006
Annulus to a square with an airfoil cavity	0.4556	1420 = 1220+200	146	0.0137	0.01	7.3014e-06	1.3696
Cube to a sphere	0.7354	20063 = 18563+1500	1304	0.0015	0.001	6.5200e-07	0.0153

For all experiments, our general approach is to prescribe a boundary deformation at a *subset* of the boundary nodes using a conformal map (or some other transformation). The RBF interpolant to the deformed boundary is then used as a proxy for the deformation map, and evaluation in the interior of the deformed domain gives us a set of interior points on that domain. We then run a few steps of our iterative smoothing algorithm to improve our mesh. In order to compare the RBF based smoothing with other smoothing techniques, we consider the Laplace smoother. At the end of each experiment in 2D, we run the same number of Laplace smoothing iterations as required by RBF-based smoothing before the stopping criterion terminates the algorithm. We also document the computational cost for each step of Algorithm 1 in Table 2 for all our experiments.

5.1 Deforming a C^1 boundary to a C^∞ boundary

For this test, we set Ω to be the unit square $[0, 1]^2$; naturally, its boundary is of limited smoothness, *i.e.*, $\partial\Omega \in C^1(\mathbb{R}^2)$. We then prescribe a deformation so that the deformed boundary $\partial\Omega'$ is a circle, and

Table 2: A summary of runtime (in seconds) for all the experiments. The table shows time-averaged per-iteration costs for the important steps in Algorithm 1 (from Section 3.2) for each of our experiments.

Step No.	C^1 boundary to C^∞ boundary	Annulus to a square with an airfoil cavity	Cube to a sphere
12	0.000621	0.000612	0.301047
13	0.000613	0.000334	0.298201
14	0.002114	0.002389	0.746076
15	0.004944	0.009064	0.226799
16	2.722284	1.563437	386.769592
18	0.000168	0.000093	14.133027
19	0.000189	0.000011	0.000015
20	0.000238	0.000018	0.000011
21	0.235500	0.382419	0.541982
23-30	0.303988	0.118030	13.342620
32	0.059021	0.152558	5.322097
33	0.005053	0.020471	0.005053
34	1.751741	3.605688	384.959461

$\partial\Omega' \in C^\infty(\mathbb{R}^2)$. The deformation map f is given component-wise by:

$$f(x, y) = \left[x \left(1 - \frac{y^2}{2} \right)^{\frac{1}{2}}, y \left(1 - \frac{x^2}{2} \right)^{\frac{1}{2}} \right]. \quad (24)$$

This conformal map has four singularities corresponding to the four corners of the square. These singularities are likely to manifest as distortions in the tessellation of the deformed domain. The results of this experiment are shown in Figure 1. Figure 1(b) shows a surface plot of the per-vertex quality measure on the undeformed domain. Figure 1(d) shows the per-vertex quality in the deformed domain obtained from RBF interpolation, and the corresponding mesh is shown in Figure 1(c). As predicted, the deformed mesh shows mild distortions corresponding to the singularities in the conformal map. Our smoothing algorithm terminates after three iterations with little improvement in per-vertex quality (Figures 1(e) and 1(f)). In fact, Figure 2 shows that the per-vertex quality *decreases* after just three iterations, which is indeed the cause for termination. We posit that this lack of improvement in quality is due to the fact that the deformed mesh is of relatively high quality to begin with. It is likely that a purely local algorithm would be able to achieve higher per-vertex quality than our quasi-local algorithm. Figure 3 shows the result of applying three iterations of Laplace smoothing to the deformed, unsmoothed mesh from Figure 1(c). Laplace smoothing, being a purely local technique, improves the quality of elements faster than the RBF-based smoothing.

5.2 Deforming an annulus into a square with an airfoil cavity

We now consider a more complicated test case. In this test, we deform a circular annulus into a square containing an airfoil. Essentially, this test transforms both inner and outer boundaries from C^∞ smoothness to C^1 smoothness. This test case shows the ability of our method to naturally handle embedded boundaries. The deformation function on the cavity boundary is the standard Joukowski conformal map [55] from circle to airfoil, while the deformation function on the outer boundary is another conformal map from a circle to a square. The results of this experiment are shown in Figure 4. Figure 4(b) shows the per-vertex quality surface plot of the undeformed mesh, while Figure 4(d) shows the same plot for the deformed mesh before smoothing. The singularity in the Joukowski map manifests as poor quality elements near the trailing edge of the airfoil. Our smoothing algorithm now runs for ten iterations before terminating; the resulting mesh is shown in Figure 4(e), and its quality is shown in Figure 4(f). The benefits of smoothing are apparent: the mesh distortions near the trailing edge have been reduced without adversely affecting the higher-quality regions. The mesh distortion near the left edge of the square has also been reduced. A study of the mesh quality as a function of the number of iterations is shown in Figure 5. While it is impossible to capture overall mesh quality with a single number, it is easy to see the improvement of quality due to the smoothing algorithm and the benefit of the termination criterion. Figure 6 shows the result of applying ten iterations

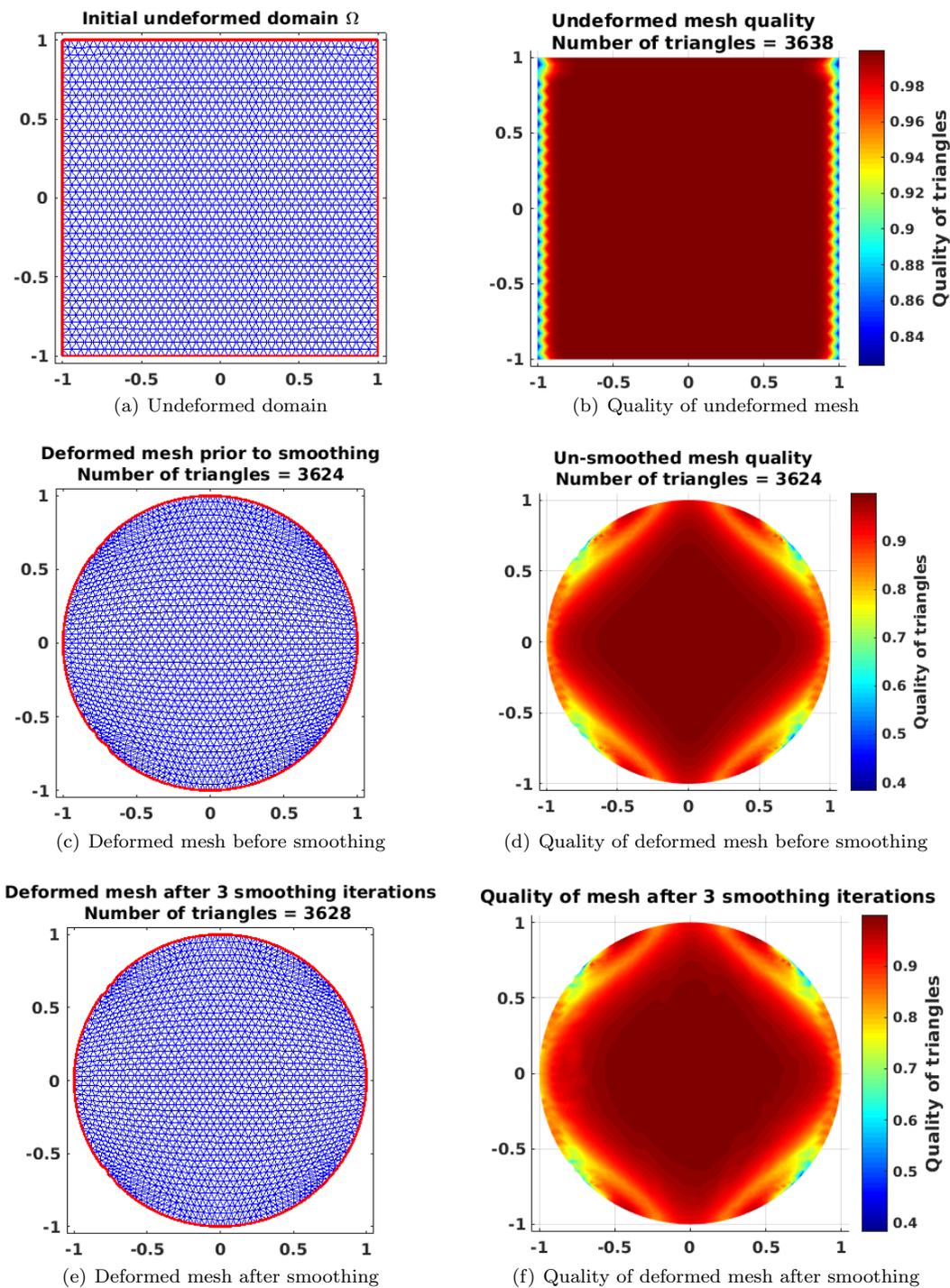


Fig. 1: Deforming a C^1 boundary to a C^∞ boundary.

of Laplace smoothing to the deformed, un-smoothed mesh from Figure 4(c). Once again, Laplace smoothing performs the smoothing faster due to its local nature.

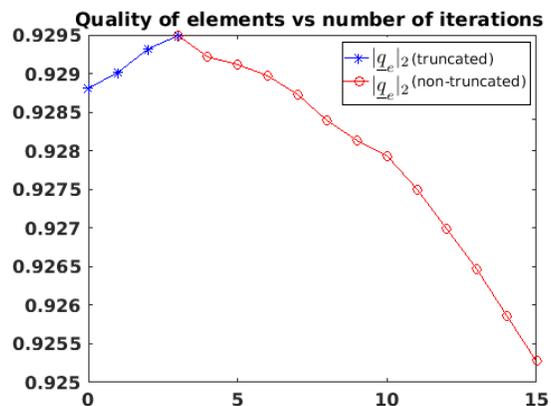


Fig. 2: Quality as a function of number of iterations when deforming a C^1 boundary to a C^∞ boundary. The blue line corresponds to the use of a stopping criterion. The red line shows the quality when no stopping criterion is used.

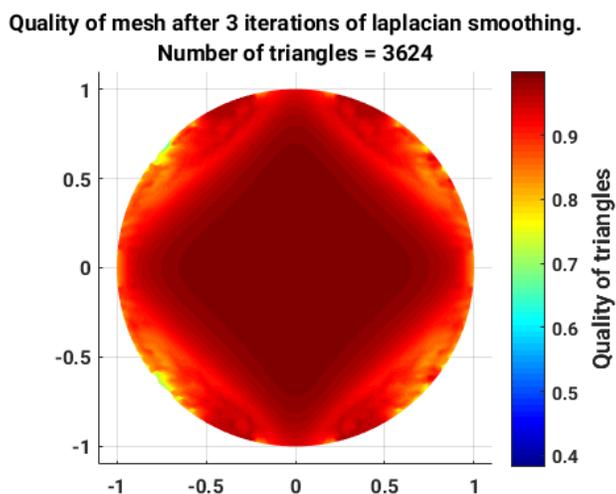


Fig. 3: Quality of the mesh after 3 iterations of Laplace smoothing applied to the original un-smoothed deformed mesh from Figure 1(c)

5.3 Deforming a cube to a sphere

One of the main advantages of RBF interpolation is that it requires no modifications for interpolating data scattered on submanifolds $M \subset \mathbb{R}^3$ [26]. Thus, our algorithm requires no modification in \mathbb{R}^3 beyond recovering the deformation map for the third coordinate. We work completely in Cartesian coordinates and do not employ special node sets on our undeformed domain boundary. We now consider the 3D analogue of the square-to-disk test by deforming the unit cube to a sphere. Once again, as in the 2D test cases, we employ a straightforward conformal map from the cube to the sphere. The undeformed domain mesh is shown in Figure 7(a), and Figures 7(b) and 7(c) show the element quality in that mesh. It is easy to see from Figures 7(b) and 7(c) that the mesh is mostly comprised of low quality elements on the boundary, and very low quality elements in the interior. We now apply the RBF interpolation and smoothing procedure to the mesh to obtain a mesh within a sphere. The resulting mesh and element quality after smoothing are shown in Figure 8. Figure 8(a) shows the curvilinear mesh obtained with the sphere. An exterior view of element quality (Figure 8(b)) shows that the mesh quality is lowest at spatial locations corresponding to singularities in the conformal map. Interestingly, both Figures 8(b) and 8(c) show that the overall element quality in the

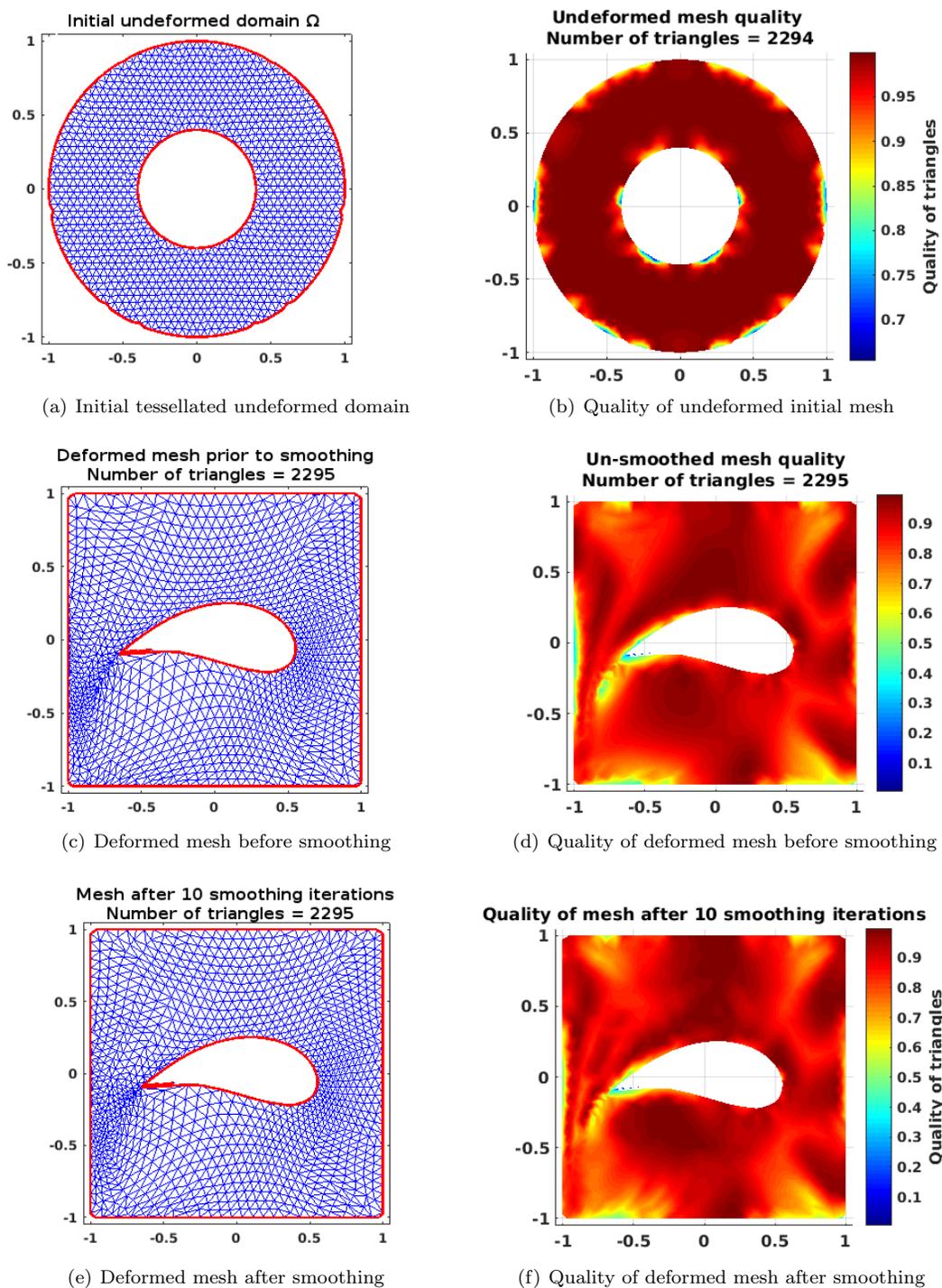


Fig. 4: Deforming an annulus into a square with an airfoil cavity.

deformed mesh is higher than in the undeformed mesh, especially in the interior. This illustrates a strength of the RBF technique in generating curvilinear meshes. It should be noted that the improvement in quality by smoothing does not invalidate any elements in the mesh. This can be easily verified by observing that no element in the deformed mesh has a negative jacobian. Finally, the last plot shows that our smoothing procedure terminates after just two iterations as further improvement is not possible. Indeed, this test case

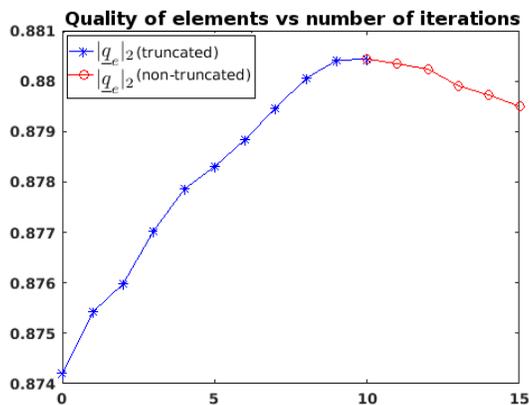


Fig. 5: Quality as a function of number of iterations when deforming an annulus into a square with an inner airfoil. The blue line corresponds to the use of a stopping criterion. The red line shows the quality when no stopping criterion is used.

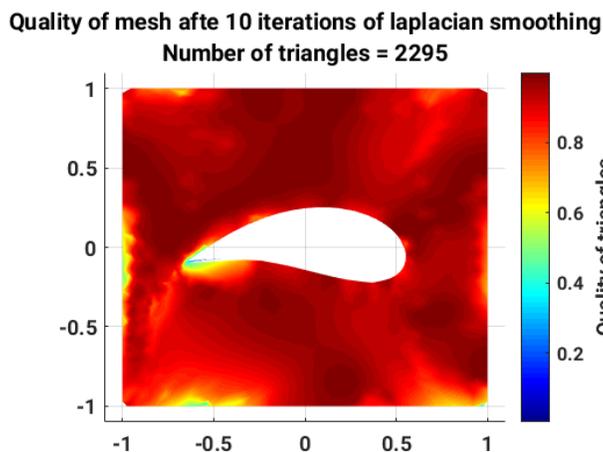


Fig. 6: Quality of the mesh after 10 iterations of Laplace smoothing applied to the original un-smoothed deformed mesh from Figure 4(c)

indicates that the quality of the undeformed mesh proves crucial in dictating the quality of the curvilinear smoothed mesh. However, this test case still illustrates that our technique is viable in 3D without any real changes to the algorithm. We also tested our technique using a high-quality mesh on the undeformed domain. In this case, we note that our smoothing algorithm did not significantly improve the quality of the mesh on the deformed domain (results not shown).

6 Discussion

The main contribution of this article is a framework for generating 2D and 3D curvilinear meshes using RBF interpolation on the domain boundary, and a quasi-local iterative algorithm for smoothing those meshes by modifying RBF shape parameters. Interestingly, the technique allows mesh generation in the interior of the domain using an approximation to the deformation map built solely on the domain boundary. Despite the maps not being harmonic functions, this technique appears to produce meshes that either preserve or improve the quality of the undeformed mesh. Our results indicate that smoothing can be beneficial, especially with meshes produced from singular deformation maps such as the Joukowski transform. Further, our algorithm

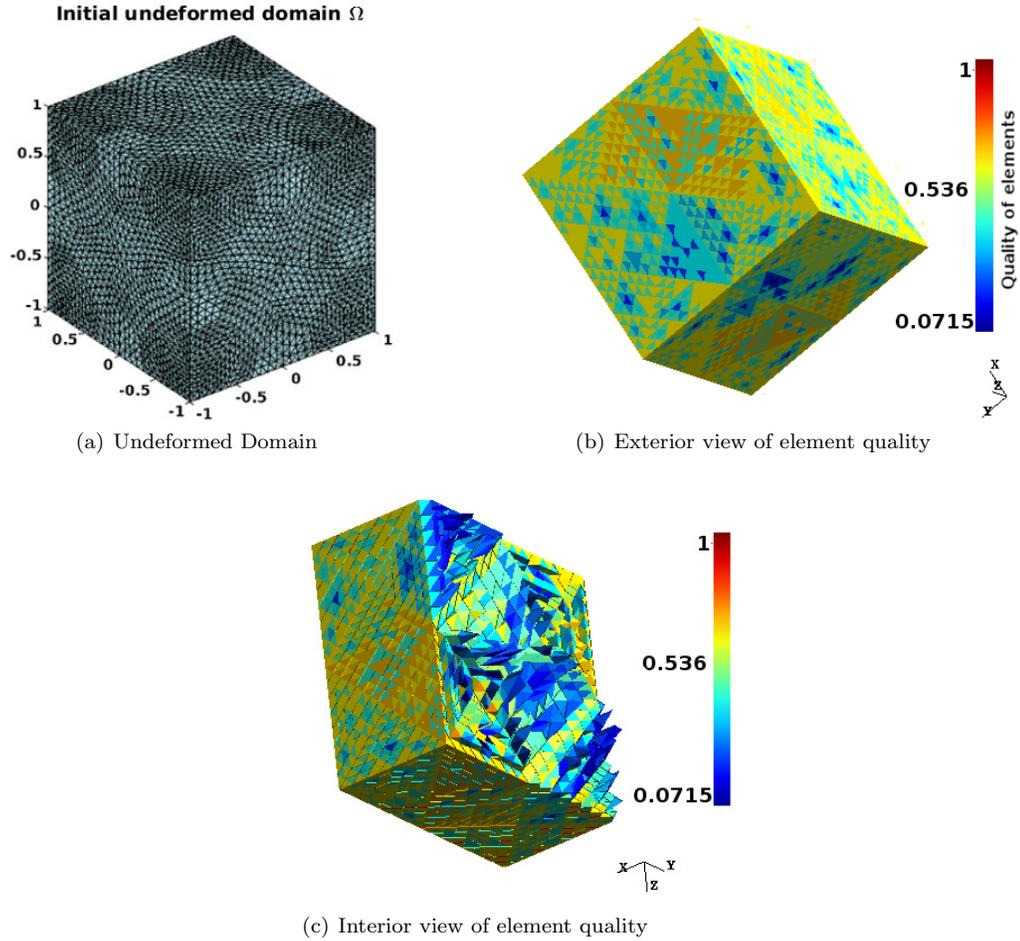


Fig. 7: Tetrahedral cube mesh and element quality.

is directly applicable to both 2D and 3D mesh generation in Cartesian coordinates due to the ability of RBF interpolants to handle scattered node sets on submanifolds of \mathbb{R}^d .

Despite its quasi-local nature, our smoothing algorithm is still global due to the use of global interpolants on the boundary. This likely limits the ability of our algorithm to handle isolated low-quality regions without adversely affecting high-quality regions. A natural approach to overcome this will be to use RBF-based Partition of Unity (RBF-PU) or RBF-based Finite Difference (RBF-FD) methods to approximate the boundary deformation map. These methods retain the advantages of global RBF interpolants— the ability to handle scattered data on submanifolds of \mathbb{R}^d , high-order convergence rates for smooth functions— but have lower costs and are more localized. Both these methods have the potential to bring down the preprocessing costs from $O(N^{1.5})$ and $O(N^2)$ to $O(N)$. Further, per-iteration evaluation costs can also be decreased to $O(N)$ from $O(N^{\frac{3}{2}})$ and $O(N^{\frac{5}{3}})$. This is an area of future research.

We remark that our algorithm is likely applicable in many scenarios beyond generating curvilinear meshes. For instance, we envision that our algorithm may be useful in remeshing particle meshes in Lagrangian methods, or in generating node sets for the meshfree or meshed solution of PDEs on time-varying domains. Finally, an open area of research is to understand how RBF interpolation on the boundary is able to recover a non-harmonic conformal map in the interior of the domain.

Acknowledgements VZ was supported by NSF OCI-1148291 and NSF IIS-1212806. VS was supported by NSF DMS-1521748. SPS was supported in part by the NIH/NIGMS Center for Integrative Biomedical Computing grant 2P41 RR0112553-12 and a grant from the ExxonMobil corporation. RMK was supported in part by DMS-1521748 and W911NF-15-1-0222.

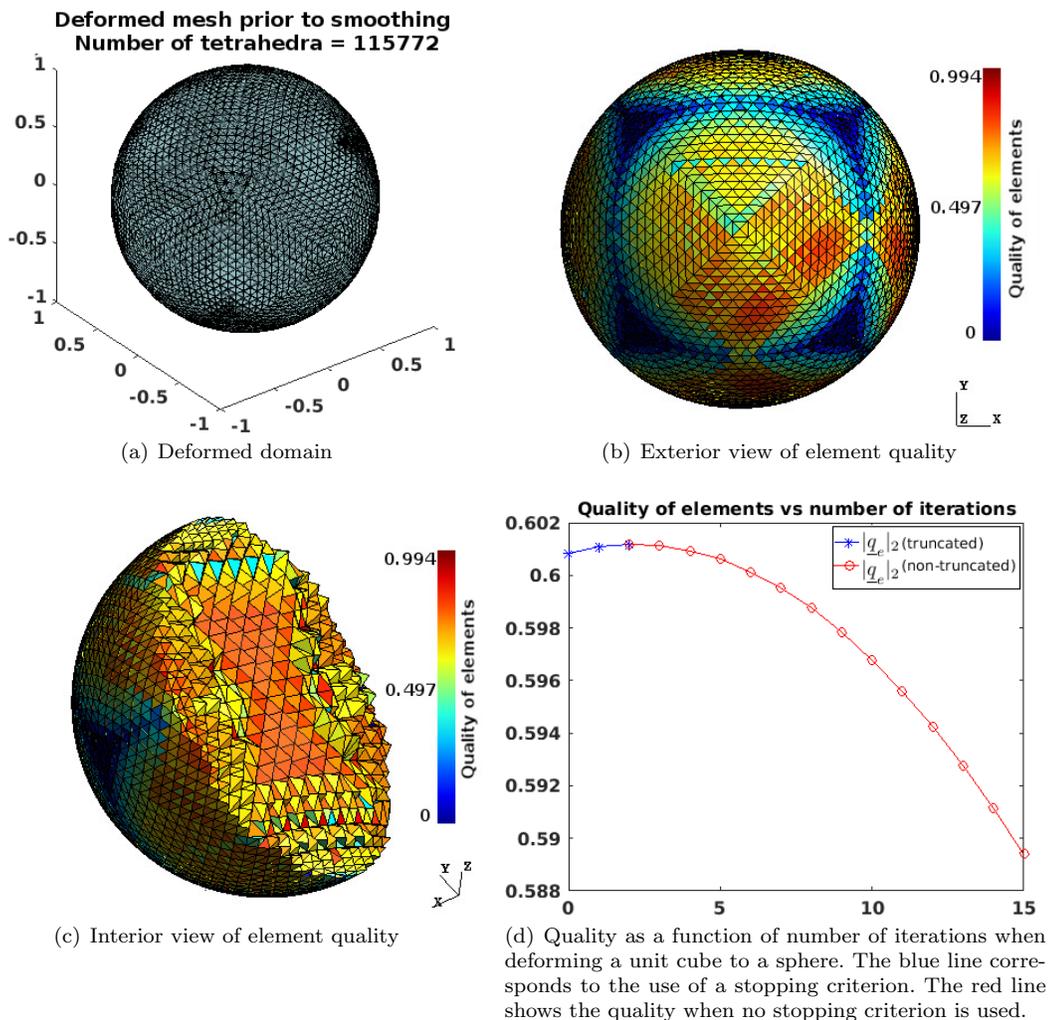


Fig. 8: Curvilinear mesh obtained by deforming the cube to the sphere.

References

1. Baker, T.: Element quality in tetrahedral meshes. 7th int. In: Conf. on Finite Element Models in Flow Problems, Huntsville, Alabama (1989)
2. Bank, R.E., Xu, J.: An algorithm for coarsening unstructured meshes. *Numerische Mathematik* **73**(1), 1–36 (1996)
3. Bhatia, R., Lawrence, K.: Two-dimensional finite element mesh generation based on stripwise automatic triangulation. *Computers & Structures* **36**(2), 309 – 319 (1990). DOI [http://dx.doi.org/10.1016/0045-7949\(90\)90131-K](http://dx.doi.org/10.1016/0045-7949(90)90131-K). URL <http://www.sciencedirect.com/science/article/pii/004579499090131K>
4. de Boer, A., van der Schoot, M.S., Bijl, H.: Mesh Deformation Based on Radial Basis Function Interpolation. *Comput. Struct.* **85**(11-14), 784–795 (2007)
5. Caendish, J.C., Field, D.A., Frey, W.H.: An approach to automatic three-dimensional finite element mesh generation. *International journal for numerical methods in engineering* **21**(2), 329–347 (1985)
6. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01*, pp. 67–76. ACM, New York, NY, USA (2001). DOI 10.1145/383259.383266. URL <http://doi.acm.org/10.1145/383259.383266>
7. Carr, J.C., Beatson, R.K., McCallum, B.C., Fright, W.R., McLennan, T.J., Mitchell, T.J.: Smooth Surface Reconstruction from Noisy Range Data. In: *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, GRAPHITE '03*, pp. 119–ff. ACM, New York, NY, USA (2003)
8. Chen, L.: Mesh smoothing schemes based on optimal delaunay triangulations. In: *Proceedings of the 13th International Meshing Roundtable, IMR 2004*, Williamsburg, Virginia, USA, September 19-22, 2004, pp. 109–120 (2004). URL <http://imr.sandia.gov/papers/abstracts/Ch317.html>

9. de Cougny, H., Georges, M., Shephard, M.: Explicit Node Point Mesh Smoothing Within the Octree Mesh Generator. SCOREC report: Scientific Computation Research Center. Program for Automated Modeling, Scientific Computation Research Center, Rensselaer Polytechnic Inst. (1990). URL <https://books.google.com/books?id=QtGHPgAACAAJ>
10. Dannelongue, H., Tanguy, P.: Three-dimensional adaptive finite element computations and applications to non-Newtonian fluids. *International journal for numerical methods in fluids* **13**(2), 145–165 (1991)
11. Driscoll, T., Fornberg, B.: Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications* **43**(3), 413–422 (2002)
12. Fasshauer, G.E.: Meshfree Approximation Methods with MATLAB. *Interdisciplinary Mathematical Sciences - Vol. 6*. World Scientific Publishers, Singapore (2007)
13. Fasshauer, G.E.: Green's functions: taking another look at kernel approximation, radial basis functions and splines, *Springer Proceedings in Mathematics*, vol. 13, p. 37–63. Springer (2011)
14. Fasshauer, G.E., McCourt, M.J.: Stable evaluation of Gaussian radial basis function interpolants. *SIAM Journal on Scientific Computing* **34**, A737–A762 (2012)
15. Field, D.: A generic Delaunay triangulation algorithm for finite element meshes. *Advances in Engineering Software and Workstations* **13**(5), 263 – 272 (1991). DOI [http://dx.doi.org/10.1016/0961-3552\(91\)90031-X](http://dx.doi.org/10.1016/0961-3552(91)90031-X). URL <http://www.sciencedirect.com/science/article/pii/096135529190031X>
16. Field, D.A.: Laplacian smoothing and delaunay triangulations. *Communications in applied numerical methods* **4**(6), 709–712 (1988)
17. Field, D.A.: Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering* **47**(4), 887–906 (2000)
18. Flyer, N., Fornberg, B., Bayona, V., Barnett, G.A.: On the role of polynomials in rbf-fd approximations: I. interpolation and accuracy. *Journal of Computational Physics* **321**, 21 – 38 (2016). DOI <http://dx.doi.org/10.1016/j.jcp.2016.05.026>. URL <http://www.sciencedirect.com/science/article/pii/S0021999116301632>
19. Fornberg, B., Larsson, E., Flyer, N.: Stable computations with Gaussian radial basis functions. *SIAM Journal on Scientific Computing* **33**(2), 869–892 (2011)
20. Fornberg, B., Lehto, E., Powell, C.: Stable calculation of Gaussian-based RBF-FD stencils. *Comp. Math. Applic.* **65**, 627–637 (2013)
21. Fornberg, B., Piret, C.: A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing* **30**, 60–80 (2007)
22. Fornberg, B., Wright, G.: Stable computation of multiquadric interpolants for all values of the shape parameter. *Comput. Math. Appl.* **48**, 853–867 (2004)
23. Fornberg, B., Zuev, J.: The Runge phenomenon and spatially variable shape parameters in RBF interpolation. *Comput. Math. Appl.* **54**, 379–398 (2007)
24. Fukuda, J., Suhara, J.: Automatic mesh generation for finite element analysis. *Advances in computational methods in structural mechanics and design* (1972)
25. Fuselier, E., Wright, G.: Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates. *SIAM Journal on Numerical Analysis* **50**(3), 1753–1776 (2012). DOI 10.1137/110821846. URL <http://epubs.siam.org/doi/abs/10.1137/110821846>
26. Fuselier, E.J., Wright, G.B.: A high-order kernel method for diffusion and reaction-diffusion equations on surfaces. *Journal of Scientific Computing* pp. 1–31 (2013). DOI 10.1007/s10915-013-9688-x. URL <http://dx.doi.org/10.1007/s10915-013-9688-x>
27. Gargallo-Peiro, A., Roca, X., Peraire, J., Sarrate, J.: Defining Quality Measures for Mesh Optimization on Parameterized CAD Surfaces. In: X. Jiao, J.C. Weill (eds.) *Proceedings of the 21st International Meshing Roundtable*, pp. 85–102. Springer Berlin Heidelberg (2013)
28. Gargallo-Peiro, A., Roca, X., Peraire, J., Sarrate, J.: Defining Quality Measures for Validation and Generation of High-Order Tetrahedral Meshes. In: J. Sarrate, M. Staten (eds.) *Proceedings of the 22nd International Meshing Roundtable*, pp. 109–126. Springer International Publishing (2014)
29. George, P., Borouchaki, H.: *Delaunay Triangulation and Meshing: Application to Finite Elements*. Butterworth-Heinemann (1998). URL <https://books.google.com/books?id=HZGfI61PSUQC>
30. Geuzaine, C., Johnen, A., Lambrechts, J., Remacle, J.F., Toulorge, T.: IDIHOM: Industrialization of High-Order Methods - A Top-Down Approach: Results of a Collaborative Research Project Funded by the European Union, 2010 - 2014, chap. The Generation of Valid Curvilinear Meshes, pp. 15–39. Springer International Publishing, Cham (2015)
31. Knupp, P.M.: Algebraic Mesh Quality Metrics. *SIAM Journal on Scientific Computing* **23**(1), 193–218 (2001). DOI 10.1137/S1064827500371499. URL <http://dx.doi.org/10.1137/S1064827500371499>
32. Larsson, E., Fornberg, B.: A numerical study of some radial basis function based solution methods for elliptic PDEs. *Computers & Mathematics with Applications* **46**(5–6), 891 – 902 (2003)
33. Larsson, E., Fornberg, B.: Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. *Comput. Math. Appl.* **49**, 103–130 (2005)
34. Macêdo, I., Gois, J.P., Velho, L.: Hermite Interpolation of Implicit Surfaces with Radial Basis Functions. In: 2009 XXII Brazilian Symposium on Computer Graphics and Image Processing, pp. 1–8 (2009)
35. Malleswaran, M., Deborah, S.A., Manjula, S., Vaidehi, V.: Integration of INS and GPS using radial basis function neural networks for vehicular navigation. In: *Control Automation Robotics Vision (ICARCV)*, 2010 11th International Conference on, pp. 2427–2430 (2010)
36. Marchandise, E., Piret, C., Remacle, J.F.: CAD and Mesh Repair with Radial Basis Functions. *J. Comput. Phys.* **231**(5), 2376–2387 (2012). DOI 10.1016/j.jcp.2011.11.033. URL <http://dx.doi.org/10.1016/j.jcp.2011.11.033>
37. Miller, T.: Optimal good-aspect-ratio coarsening for unstructured meshes. In: *SODA: ACM-SIAM Symposium on Discrete Algorithms* (1997)

38. Möller, P., Hansbo, P.: On advancing front mesh generation in three dimensions. *International Journal for Numerical Methods in Engineering* **38**(21), 3551–3569 (1995). DOI 10.1002/nme.1620382102. URL <http://dx.doi.org/10.1002/nme.1620382102>
39. Moxey, D., Ekelschot, D., Keskin, Ü., Sherwin, S., Peiró, J.: High-order curvilinear meshing using a thermo-elastic analogy. *Computer-Aided Design* **72**, 130 – 139 (2016). 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation
40. Moxey, D., Green, M., Sherwin, S., Peiró, J.: An isoparametric approach to high-order curvilinear boundary-layer meshing. *Computer Methods in Applied Mechanics and Engineering* **283**, 636–650 (2015). Cited By 3
41. P.-O. Persson, J.P.: Curved mesh generation and mesh refinement using Lagrangian solid mechanics. Proceedings of the 47th AIAA Aerospace Sciences Meeting, Orlando, Florida, American Institute of Aeronautics and Astronautics, Inc. pp. 949:1–11 (2009)
42. Parthasarathy, V., Graichen, C., Hathaway, A.: A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design* **15**(3), 255–261 (1994)
43. Parthasarathy, V., Kodiyalam, S.: A constrained optimization approach to finite element mesh smoothing. *Finite Elements in Analysis and Design* **9**(4), 309–320 (1991)
44. Perronnet, A.: Triangulation par arbre-4 de triangles équilatéraux et maximisation de la qualité. Tech. Rep. R 92015-Vol. 11 ; fasc. 3, Université Pierre et Marie Curie (Paris) (1992). URL <http://opac.inria.fr/record=b1031284>
45. Persson, P.O., Strang, G.: A Simple Mesh Generator in MATLAB. *SIAM Review* pp. 329–345 (2004)
46. Remacle, J.F., Toulorge, T., Lambrechts, J.: Proceedings of the 21st International Meshing Roundtable, chap. Robust Untangling of Curvilinear Meshes, pp. 71–83. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
47. Sastry, S.P., Shontz, S.M., Vavasis, S.A.: A Log-barrier Method for Mesh Quality Improvement and Untangling. *Engineering with Computers* **30**(3), 315–329 (2014)
48. Sastry, S.P., Zala, V., Kirby, R.M.: Thin-plate-Spline Curvilinear Meshing on a Calculus-of-Variations Framework. *Procedia Engineering* **124**, 135 – 147 (2015). 24th International Meshing Roundtable
49. Savitha, R., Suresh, S., Sundararajan, N.: A Fully Complex-Valued Radial Basis Function Network and its Learning Algorithm. *International Journal of Neural Systems* **19**(04), 253–267 (2009). PMID: 19731399
50. Schaback, R.: Multivariate interpolation by polynomials and radial basis functions. *Constr. Approx.* **21**, 293–317 (2005)
51. Shankar, V., Wright, G., Kirby, R., Fogelson, A.: A Radial Basis Function (RBF)-Finite Difference (FD) Method for Diffusion and Reaction-Diffusion Equations on Surfaces. *Journal of Scientific Computing* **63**(3), 745–768 (2015). DOI 10.1007/s10915-014-9914-1
52. Shankar, V., Wright, G.B., Fogelson, A.L., Kirby, R.M.: A Radial Basis Function (RBF)-Finite Difference Method for the Simulation of Reaction-Diffusion Equations on Stationary Platelets within the Augmented Forcing Method. *International Journal for Numerical Methods in Fluids* **75**(1), 1–22 (2014). DOI 10.1002/fld.3880. URL <http://dx.doi.org/10.1002/fld.3880>
53. Staten, M.L., Owen, S.J., Shontz, S.M., Salinger, A.G., Coffey, T.S.: Proceedings of the 20th International Meshing Roundtable, chap. A Comparison of Mesh Morphing Methods for 3D Shape Optimization, pp. 293–311. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
54. Toulorge, T., Geuzaine, C., Remacle, J.F., Lambrechts, J.: Robust Untangling of Curvilinear Meshes. *J. Comput. Phys.* **254**, 8–26 (2013)
55. Tsien, H.S.: Symmetrical Joukowski airfoils in shear flow. *Quarterly of Applied Mathematics* **1**(2), 130–148 (1943)
56. Turner, M., Peir, J., Moxey, D.: A Variational Framework for High-order Mesh Generation . *Procedia Engineering* **163**, 340 – 352 (2016). DOI <http://dx.doi.org/10.1016/j.proeng.2016.11.069>. URL <http://www.sciencedirect.com/science/article/pii/S1877705816333781>. 25th International Meshing Roundtable
57. Watabayashi, G., Galt, J.: An optimized triangular mesh system from random points. *Numerical Grid Generation in Computational Fluid Dynamics* pp. 437–438 (1986)