Álvarez Román, D., González Rodríguez, P. y Moscoso Castro, M. (2018). A Closed-Form Formula for the RBF-Based Approximation of the Laplace–Beltrami Operator. *Journal of Scientific Computing,* 77 (2), pp. 1115-1132.

DOI: https://doi.org/10.1007/s10915-018-0739-1

A Closed-Form Formula for the RBF-Based Approximation of the Laplace–Beltrami Operator

Diego Álvarez[1] · **Pedro González-Rodríguez**[1] · **Miguel Moscoso**[1]

Received: 2 January 2018 / Revised: 11 May 2018 / Accepted: 14 May 2018

Abstract In this paper we present a method that uses radial basis functions to approximate the Laplace–Beltrami operator that allows to solve numerically diffusion (and reaction–diffusion) equations on smooth, closed surfaces embedded in $\mathbb{R}^3$. The novelty of the method is in a closed-form formula for the Laplace–Beltrami operator derived in the paper, which involve the normal vector and the curvature at a set of points on the surface of interest. An advantage of the proposed method is that it does not rely on the explicit knowledge of the surface, which can be simply defined by a set of scattered nodes. In that case, the surface is represented by a level set function from which we can compute the needed normal vectors and the curvature. The formula for the Laplace–Beltrami operator is exact for radial basis functions and it also depends on the first and second derivatives of these functions at the scattered nodes that define the surface. We analyze the converge of the method and we present numerical simulations that show its performance. We include an application that arises in cardiology.

Keywords Radial basis functions · Surface Laplacian · Surface PDE · Cardiology

Mathematics Subject Classification 65D05 · 35R01 · 58J05

✉ Diego Álvarez
jdiego@math.uc3m.es

Pedro González-Rodríguez
pgonzale@ing.uc3m.es

Miguel Moscoso
moscoso@math.uc3m.es

[1] Departamento de Matemáticas, Universidad Carlos III de Madrid, Avenida de la Universidad 30, 28911 Leganés, Spain

1 Introduction

Many problems in sciences and engineering require the solution to partial differential equations (PDEs) on a curved surface. These equations are formulated in terms of differential operators acting on functions defined on the surface, such as the Laplace–Beltrami operator. Problems of interest include image processing [25], phase changes of a material on a surface [35], domain formation in vesicles [2], pattern formation on a growing biological surface [3], modeling of surface active agents [20], ice accretion and water flow on cold surfaces such as aircrafts [28], and restore a damaged pattern on a surface such as a vase [5].

Analytical solutions to PDEs on surfaces are usually not possible and, hence, there is a great interest in developing numerical methods for these problems. Surface finite elements, for example, use a polyhedral approximation of the curved surface based on triangulation [10]. One of its advantages is that only the surface is discretized, rather than the entire volume, thereby keeping the dimensionality of the original problem. Its main drawback is the expensive task of generating a computational mesh over the surface, which is a burden if additional nodes need to be considered in the mesh. Other methods solve the PDE on a flat surface instead, and map the solution through a suitable parameterization of the surface [34]. However, a parameterization of an arbitrary surface is not always practical, and it may introduce artificial singularities (e.g. spherical coordinates introduce singularities at the poles) [11]. Embedding techniques, on the other hand, solve the PDE in a small region near the surface [19,24,30]. They use standard Cartesian grid methods and, thus, coordinate singularities are avoided. These methods, however, may give problems when one extends the boundary conditions or the initial data on the surface to the narrow band surrounding the surface [30].

On the other hand, RBF methods [21,22] have been successfully used to find the solution of PDEs in various geometries in $\mathbb{R}^2$ and $\mathbb{R}^3$ because they are geometrically flexible, easy to implement, and can be highly accurate. For example, in [12,13,38] global RBF-based spatial discretizations are used to solve PDEs that model a variety of phenomena in geosciences, such as mantle flows in spherical shells. In [29], the author combines global RBF and orthogonal gradient methods to solve PDEs on more general surfaces. In [17], diffusion and reaction–diffusion PDEs on surfaces arising in biology and chemistry are also solved with a global RBF method. In that work, the surface gradient is approximated using collocation, while the surface Laplacian is approximated by applying the estimated surface gradient twice. Also, local RBF methods, where the solution is approximated within a small influence domain instead of a global one, have been investigated to solve PDEs on curved surfaces [32]. Their computational cost is much lower but the spectral accuracy associated to the global version is lost. Local RBF methods are also known as RBF finite difference methods because RBFs are used to create weights for finite difference like stencils with scattered nodes [4,33,36].

In this paper, we propose a global RBF approach to solve PDEs on surfaces only defined by a cloud of points. An explicit representation of the surface is not needed. In the propose method, the surface is represented by a level set function found by standard RBF interpolation. This allows us to estimate the normal vector and the curvature at any point on the surface, which are needed for the closed-form formula of the surface Laplacian operator. The formula is exact for functions whose values depend only on the distance from an origin and, hence, the formula is exact for RBFs. Error bounds for the approximation of surface Laplacian applied to more general functions are also given. To show the convergence of the proposed method we also compute numerically the spherical Laplacian of a function whose analytical solution is known and, thus, we display the errors as function of the RBF shape parameter and of the

number of points defining the surface. We also apply the method to a non-linear system of PDEs that arise in cardiology, thus showing an application to a realistic problem.

The paper is organized as follows. In Sect. 2, we review the basic tools needed for RBF interpolation and for representing surfaces using the level set formalism. In Sect. 3, we derive formulas for the surface gradient and the surface Laplacian, and we give error bounds for these formulas. In Sect. 4, we show numerical experiments that illustrate the performance of our method. Finally, Sect. 5 contains our conclusions. In "Appendix A" we give the details for the special case of Laplacian of an RBF on the surface of a unit sphere.

2 RBF Interpolation and RBF Surface Modeling

In this section, we review the basic concepts and definitions for RBF interpolation of functions defined on surfaces in $\mathbb{R}^3$, and we summarize the level set formalism used to reconstruct surfaces from point-clouds in $\mathbb{R}^3$. Both are key to the method explained in the next section.

2.1 RBF Interpolation

Let $f : \Sigma \to \mathbb{R}$ be a continuous function sampled on a set of scattered points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ on a surface Σ embedded in $\mathbb{R}^3$. The basic RBF interpolant of f takes the form

$$s_f(\mathbf{x}) = \sum_{i=1}^{N} c_i \phi(r_i(\mathbf{x})), \tag{1}$$

where $r_i(\mathbf{x}) = \|\mathbf{r}_i(\mathbf{x})\| = \|\mathbf{x} - \mathbf{x}_i\|$ is the distance from a point $\mathbf{x}$ on the surface to the RBF center $\mathbf{x}_i$. Here, $\|\cdot\|$ is the Euclidean norm, and $\phi(r_i(\mathbf{x}))$ is some radial function which often contains a shape parameter ϵ that affects the accuracy of the RBF interpolation (1). For given data values $f_i = f(\mathbf{x}_i), i = 1 \ldots, N$, the interpolation coefficients c_i are obtained by solving the linear system

$$A\,\boldsymbol{c} \;=\; \mathbf{f}, \tag{2}$$

where the entries of the $N \times N$ interpolation matrix A are $a_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$. In (2), $\boldsymbol{c} = [\, c_1 \; c_2 \; \cdots \; c_N \,]^T$ and $\mathbf{f} = [\, f_1 \; f_2 \; \cdots \; f_N \,]^T$ are N-dimensional column vectors. This equation implies that (1) interpolates $f(\mathbf{x})$ at the set of points $\mathbf{X}$. The interpolation matrix A is dense and symmetric. A sufficient condition for the nonsingularity of this matrix is strictly positive definiteness [27]. Furthermore, it is well known that large values of the shape parameter ϵ lead to well-conditioned linear systems, although the resulting approximation may be inaccurate. On the other hand, small values of ϵ lead to accurate results but the condition number of the interpolation matrix A is large and, hence, the interpolation coefficients c_i may diverge [9,34]. To overcome this problem, the interpolation coefficients in the limit $\epsilon \to 0$ are computed in [18,23] by means of the Laurent series of the inverse of the interpolation matrix A.

For a more detailed discussion on RBF interpolation see, for example, [6].

2.2 Modeling Surfaces with RBFs

In this paper, we assume that the surfaces are defined by point-clouds. Next, we summarize the level set formalism used for reconstructing surfaces in these situations. The goal is to find a continuously differentiable level-set function $\Psi : \mathbb{R}^3 \to \mathbb{R}$, such that its zero level-set

$$\Sigma = \{\mathbf{x} \in \mathbb{R}^3 : \Psi(\mathbf{x}) = 0\} \tag{3}$$

specifies the sought surface. The main advantage of this approach is that one can represent the surfaces on a Cartesian grid without having to parameterize them. This makes the level-set method very flexible when, for example, the surfaces change with time. Furthermore, once the level set function Ψ has been found, the unit normal vector and the curvature can be easily computed at any point on the surface; see Sect. 3.3.

Given a set of points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ on the surface, we seek a smooth function Ψ that satisfies the interpolant conditions $\Psi(\mathbf{x}_i) = 0, i = 1, \ldots, N$. In order to avoid the trivial solution $\Psi(\mathbf{x}) = 0$, M off-surface points $\{\mathbf{x}_i\}_{i=N+1}^{N+M}$ are added to the data, such that $\Psi(\mathbf{x}_i) \neq 0, i = N+1, \ldots, N+M$. We assign positive or negative values to $\Psi(\mathbf{x}_i)$ depending whether the nodes are on one side or the other of the surface. Thus, we model the surface implicitly with a level-set function Ψ such that

$$\begin{array}{lll} \Psi(\mathbf{x}_i) = 0, & i = 1, \ldots, N & \text{(on-surface nodes)}, \\ \Psi(\mathbf{x}_i) \neq 0, & i = N+1, \ldots, N+M & \text{(off-surface nodes)}. \end{array} \tag{4}$$

Then, the interpolant

$$s_\Psi(\mathbf{x}) = \sum_{i=1}^{N+M} d_i \, \phi(r_i(\mathbf{x})) \tag{5}$$

is found by imposing $s_\Psi(\mathbf{x}_i) = \Psi(\mathbf{x}_i), i = 1, \ldots, N+M$. The sought surface Σ is then defined by the zero-set of the interpolant (5), that is, the surface consists of all points $\mathbf{x} \in \mathbb{R}^3$ satisfying $s_\Psi(\mathbf{x}) = 0$. The coefficients d_i in (5) are found by solving (2) with $\mathbf{f} = [\,\Psi(\mathbf{x}_1)\ \Psi(\mathbf{x}_2)\ \cdots\ \Psi(\mathbf{x}_{N+M})\,]^T$.

The problem of placing the off-surface nodes and the assignment of the values $\Psi(\mathbf{x}_i), i = N+1, \ldots, N+M$ may not be trivial. A common practice is to generate pairs of extra points, at either side of the surface, at a small distance along the normals of the surface, and to give values to $\Psi(\mathbf{x}_i)$ proportional to the distance to the closest point on the surface. If the normal vectors are not known explicitly, then they have to be estimated, a task which might be not easy. Our experience have shown, however, that if the surface is smooth enough it suffices to just add a few off-surface nodes at each side of the surface with values $\Psi(\mathbf{x}_i) = \pm 1$ depending whether they are at one side or the other to the surface. In the numerical experiments shown below we have placed eight extra points outside the surface, and one extra point inside of it. The exterior points are placed in the vertices of a box that contains the surface. The interior point is located at the center of the box. We refer the reader to [7] for more details about the reconstruction and representation of surfaces with RBFs.

3 Surface Derivative Operators

Next, we derive expressions for the surface gradient and the surface Laplacian operators of the approximant of any given function expressed in terms of RBFs. We assume that the surface embedded in $\mathbb{R}^3$ is smooth enough and has no boundaries. All the expressions are given in Cartesian coordinates.

3.1 Surface Gradient Operator

Let the unit outward vector perpendicular to a surface Σ at a given point $\mathbf{x}$ be denoted by $\mathbf{n} = \mathbf{n}(\mathbf{x})$. To make the notation simpler we will omit the dependence of the normal vector on the position $\mathbf{x}$. Then, the surface gradient operator ∇_Σ is given by

$$\nabla_\Sigma(\cdot) = \nabla(\cdot) - (\mathbf{n} \cdot \nabla(\cdot))\mathbf{n} = \nabla(\cdot) - \mathrm{D}_\mathbf{n}(\cdot)\mathbf{n}, \tag{6}$$

where ∇ denotes the standard gradient operator in $\mathbb{R}^3$, and $\mathrm{D}_\mathbf{n}(\cdot) = \mathbf{n} \cdot \nabla(\cdot)$ denotes the directional derivative along the normal vector $\mathbf{n}$.

If the surface gradient operator (6) acts on an RBF $\phi(r_i(\mathbf{x}))$, whose values only depend on the distance $r_i(\mathbf{x})$ to the node $\mathbf{x}_i$, then we obtain using the chain rule

$$\nabla_\Sigma \phi(r_i) = \nabla\phi(r_i) - \mathrm{D}_\mathbf{n}(\phi(r_i))\,\mathbf{n} = (\nabla r_i - \mathrm{D}_\mathbf{n}(r_i)\mathbf{n})\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i}. \tag{7}$$

In (7), and in all that follows, we use $r_i = r_i(\mathbf{x})$ to simplify the notation. Taking into account that

$$\nabla r_i = \frac{\mathbf{r}_i}{r_i} \quad \text{and} \quad \mathrm{D}_\mathbf{n}(r_i) = \frac{\mathbf{r}_i \cdot \mathbf{n}}{r_i}, \tag{8}$$

we write the surface gradient of an RBF as

$$\nabla_\Sigma \phi(r_i) = (\mathbf{r}_i - (\mathbf{r}_i \cdot \mathbf{n})\mathbf{n})\,\frac{1}{r_i}\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i}. \tag{9}$$

With this expression we are able to approximate the surface gradient of any function f sampled at a set of points on the surface. Indeed, using the RBF interpolation (1) we readily obtain

$$\nabla_\Sigma f(\mathbf{x}) \approx \nabla_\Sigma s_f(\mathbf{x}) = \sum_{i=1}^{N} d_i\,\nabla_\Sigma \phi(r_i(\mathbf{x})), \tag{10}$$

with coefficients d_i found by solving (2).

Given a set of scattered points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ on the surface, each component $\xi = x$, y or z of the gradient vectors at those points can be written in matrix form as

$$\begin{bmatrix} \nabla^\xi_\Sigma f(\mathbf{x}_1) \\ \nabla^\xi_\Sigma f(\mathbf{x}_2) \\ \vdots \\ \nabla^\xi_\Sigma f(\mathbf{x}_N) \end{bmatrix} \approx G^\xi_\Sigma\,\mathbf{f}, \tag{11}$$

where the matrix

$$G^\xi_\Sigma = B^\xi\,A^{-1} \tag{12}$$

is the product of the matrix $B^\xi = [B]^\xi_{ij} = \nabla^\xi_\Sigma \phi(r_i(\mathbf{x}_j))$ and the inverse of the interpolation matrix A. The entries of the matrix B^ξ are just the ξ component of the surface gradient vectors of the RBF functions $\phi(r_i)$ evaluated at the points $\mathbf{x}_j$. Equation (12) is also given in [17].

3.2 Surface Laplacian Operator

We now derive an RBF-based closed-form expression for the surface Laplacian of a function defined on a surface. This operator can be computed as the surface divergence of the surface gradient, that is,

$$\Delta_\Sigma(\cdot) = \mathrm{div}_\Sigma\left(\nabla_\Sigma(\cdot)\right) = \nabla_\Sigma \cdot \nabla_\Sigma(\cdot). \tag{13}$$

Thus, using (6) the surface Laplacian of a function defined on a surface Σ can be written as

$$\begin{aligned}\Delta_\Sigma(\cdot) &= \nabla_\Sigma \cdot \nabla(\cdot) - \nabla_\Sigma \cdot (\mathrm{D}_{\mathbf{n}}(\cdot)\mathbf{n}) \\ &= \Delta(\cdot) - \mathrm{D}_{\mathbf{n}}(\nabla(\cdot)) \cdot \mathbf{n} - \nabla_\Sigma \cdot (\mathrm{D}_{\mathbf{n}}(\cdot)\,\mathbf{n}),\end{aligned} \tag{14}$$

where the first term represents the standard Laplacian operator in $\mathbb{R}^3$.

Next, we apply (14) to an RBF $\phi(r_i)$ whose center is at $\mathbf{x}_i$. For the first term in (14) we obtain

$$\Delta\phi(r_i) = \Delta r_i \,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + \|\nabla r_i\|^2 \,\frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2} = \frac{2}{r_i}\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + \frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2}, \tag{15}$$

where we have used $\|\nabla r_i\|^2 = 1$, and

$$\Delta r_i = \nabla \cdot \frac{\mathbf{r}_i}{r_i} = \frac{r_i \nabla \cdot \mathbf{r}_i - \mathbf{r}_i \cdot \nabla r_i}{r_i^2} = \frac{3r_i - \dfrac{\mathbf{r}_i \cdot \mathbf{r}_i}{r_i}}{r_i^2} = \frac{2}{r_i}. \tag{16}$$

For the second term in (14) we obtain

$$\begin{aligned}\mathrm{D}_{\mathbf{n}}(\nabla\phi(r_i)) \cdot \mathbf{n} &= \mathrm{D}_{\mathbf{n}}\left(\nabla r_i\right) \cdot \mathbf{n}\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + \nabla r_i \cdot \mathbf{n}\,\mathrm{D}_{\mathbf{n}}(r_i)\frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2} \\ &= \left(1 - \frac{(\mathbf{n}\cdot\mathbf{r}_i)^2}{r_i^2}\right)\frac{1}{r_i}\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + \left(\frac{(\mathbf{n}\cdot\mathbf{r}_i)^2}{r_i^2}\right)\frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2},\end{aligned} \tag{17}$$

where we have used the expression (8), the fact that $\mathrm{D}_{\mathbf{n}}(\mathbf{r}_i) = \mathbf{n}$, and

$$\mathrm{D}_{\mathbf{n}}(\nabla r_i) \cdot \mathbf{n} = \frac{1}{r_i^2}\left(r_i \mathrm{D}_{\mathbf{n}}(\mathbf{r}_i) - \mathrm{D}_{\mathbf{n}}(r_i)\mathbf{r}_i\right) \cdot \mathbf{n} = \left(1 - \frac{(\mathbf{n}\cdot\mathbf{r}_i)^2}{r_i^2}\right)\frac{1}{r_i}.$$

Finally, for the third term in (14) we obtain

$$\nabla_\Sigma \cdot (\mathrm{D}_{\mathbf{n}}(\phi(r_i))\mathbf{n}) = \mathrm{D}_{\mathbf{n}}(\phi(r_i))\nabla_\Sigma \cdot \mathbf{n} + \mathbf{n} \cdot \nabla_\Sigma\left(\mathrm{D}_{\mathbf{n}}(\phi(r_i))\right) = \kappa\,\frac{\mathbf{n}\cdot\mathbf{r}_i}{r_i}\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i}, \tag{18}$$

where $\kappa = \kappa(\mathbf{x}) = \nabla_\Sigma \cdot \mathbf{n}(\mathbf{x})$ is the curvature of the surface Σ at position $\mathbf{x}$. Note that $\mathbf{n} \cdot \nabla_\Sigma(\cdot)$ vanishes because it is the scalar product of two orthogonal vectors.

Using these results we write the surface Laplacian of an RBF as

$$\Delta_\Sigma\phi(r_i) = \left(1 + \frac{(\mathbf{n}\cdot\mathbf{r}_i)^2}{r_i^2} - \kappa\,(\mathbf{n}\cdot\mathbf{r}_i)\right)\frac{1}{r_i}\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + \left(1 - \frac{(\mathbf{n}\cdot\mathbf{r}_i)^2}{r_i^2}\right)\frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2}. \tag{19}$$

Hence, $\Delta_\Sigma\phi(r_i(\mathbf{x}))$ only depends on the first and second derivatives of the RBF at the point $\mathbf{x}$, and on the normal vector and the curvature of the surface at that point. We derive expressions for the normal vector and the curvature in the next subsection. Formula (19) is an important result of the paper, and it simplifies to

$$\Delta_{\mathbb{S}^2}\phi(r_i) = \frac{1}{4}\left(\frac{4-3r_i^2}{r_i}\,\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + (4-r_i^2)\,\frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2}\right) \tag{20}$$

for the case in which Σ is the surface of the unit sphere $\mathbb{S}^2$. Equation (20) agrees with the result given in [15] for the same case (see "Appendix A" for details).

As with the surface gradient operator, it is now straightforward to approximate the surface Laplacian of a function defined on a surface using formula (19). Proceeding in a similar way as for the surface gradient operator we obtain that

$$\Delta_\Sigma f(\mathbf{x}) \approx \Delta_\Sigma s_f(\mathbf{x}) = \sum_{i=1}^{N} d_i\, \Delta_\Sigma \phi(r_i(\mathbf{x})), \tag{21}$$

with the coefficients d_i found from (2). The surface Laplacian at the points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ can be written in matrix form as

$$\begin{bmatrix} \Delta_\Sigma f(\mathbf{x}_1) \\ \Delta_\Sigma f(\mathbf{x}_2) \\ \vdots \\ \Delta_\Sigma f(\mathbf{x}_N) \end{bmatrix} \approx L_\Sigma\, \mathbf{f}, \tag{22}$$

where the matrix

$$L_\Sigma = C\, A^{-1} \tag{23}$$

is the product of the matrix $C = [C]_{ij} = \Delta_\Sigma \phi(r_i(\mathbf{x}_j))$ and the inverse of the interpolation matrix A. The entries of the matrix C are just the surface Laplacian of the RBF functions $\phi(r_i)$ at the points $\mathbf{x}_j$.

3.3 Computation of the Normal Vector and the Curvature

Next, we compute the normal vector $\mathbf{n} = \mathbf{n}(\mathbf{x})$ and the curvature term $\kappa = \kappa(\mathbf{x})$ of a surface defined by a set of scattered points. Let us assume that our surface Σ is given by the zero-set of a function $\Psi(\mathbf{x})$, which has be found by RBF interpolation solving (5). Once the function

$$\Psi(\mathbf{x}) = \sum_{i=1}^{N+M} d_i\, \phi(r_i(\mathbf{x})) \tag{24}$$

is known, the outward-pointing unit normal vector to the surface Σ at a point $\mathbf{x}$ is simply given by

$$\mathbf{n} = \frac{\nabla\Psi}{\|\nabla\Psi\|}, \tag{25}$$

where the gradient of Ψ can be computed as

$$\nabla\Psi = \sum_{i=1}^{N+M} d_i\, \frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i}\,\frac{\mathbf{r}_i}{r_i}. \tag{26}$$

On the other hand, the curvature at a point $\mathbf{x}$ can be obtained from

$$\kappa = \nabla_\Sigma \cdot \mathbf{n} = \nabla\cdot\mathbf{n} - \mathrm{D}_{\mathbf{n}}(\mathbf{n})\cdot\mathbf{n}. \tag{27}$$

To find an explicit formula for κ we use (25) to express the two terms in (27) as function of Ψ. Indeed, using that

$$\nabla \cdot \mathbf{n} = \frac{\|\nabla\Psi\|\,\nabla\cdot\nabla\Psi - \nabla\Psi\cdot\nabla\|\nabla\Psi\|}{\|\nabla\Psi\|^2} = \frac{1}{\|\nabla\Psi\|}\left(\Delta\Psi - \mathrm{D}_{\mathbf{n}}\left(\|\nabla\Psi\|\right)\right), \tag{28}$$

and

$$\begin{aligned}\mathrm{D}_{\mathbf{n}}(\mathbf{n})\cdot\mathbf{n} &= \frac{\|\nabla\Psi\|\mathrm{D}_{\mathbf{n}}\left(\nabla\Psi\right) - \mathrm{D}_{\mathbf{n}}\left(\|\nabla\Psi\|\right)\nabla\Psi}{\|\nabla\Psi\|^2}\cdot\frac{\nabla\Psi}{\|\nabla\Psi\|}\\ &= \frac{1}{\|\nabla\Psi\|}\left(\mathrm{D}_{\mathbf{n}}\left(\nabla\Psi\right)\cdot\mathbf{n} - \mathrm{D}_{\mathbf{n}}\left(\|\nabla\Psi\|\right)\right),\end{aligned} \tag{29}$$

Equation (27) can be written as

$$\kappa = \frac{1}{\|\nabla\Psi\|}\left(\Delta\Psi - \mathrm{D}_{\mathbf{n}}\left(\nabla\Psi\right)\cdot\mathbf{n}\right). \tag{30}$$

Finally, using (15) and (17) we obtain

$$\kappa = \frac{1}{\|\nabla\Psi\|}\sum_{i=1}^{N+M} d_i\left(\left(1+\frac{(\mathbf{n}\cdot\mathbf{r}_i)^2}{r_i^2}\right)\frac{1}{r_i}\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + \left(1-\frac{(\mathbf{n}\cdot\mathbf{r}_i)^2}{r_i^2}\right)\frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2}\right), \tag{31}$$

with $\nabla\Psi$ given by (26). This formula for the curvature depends on the normal vectors and on the first and second derivatives of the RBFs at the points that define the surface. Its accuracy only relies on the quality of the interpolation.

3.4 Error Estimates for the Approximations of the Surface Derivatives

In this section, we give error bounds for the RBF approximations to the surface differential operators obtained previously. Given a smooth closed surface Σ in $\mathbb{R}^3$, and a function $f : \Sigma \to \mathbb{R}$, we denote the exact surface gradient and the exact surface Laplacian of f on Σ as $\nabla_\Sigma f(\mathbf{x})$ and $\Delta_\Sigma f(\mathbf{x})$, respectively. On the other hand, let $\nabla_\Sigma s_f(\mathbf{x})$ and $\Delta_\Sigma s_f(\mathbf{x})$ be the approximations to these differential operators given by (10) and (21). Then, following [16] we give bounds for the errors

$$\|\nabla_\Sigma f(\mathbf{x}) - \nabla_\Sigma s_f(\mathbf{x})\|_{L_\infty(\Sigma)} \qquad \text{and} \qquad \|\Delta_\Sigma f(\mathbf{x}) - \Delta_\Sigma s_f(\mathbf{x})\|_{L_\infty(\Sigma)}, \tag{32}$$

where $\|\cdot\|_{L_\infty(\Sigma)}$ is the L_∞-norm over Σ. Since (10) and (21) only depend on the RBF interpolation, it is natural to expect that (32) only depends on the accuracy of the RBF interpolation as well.

In [16], the authors give Sobolev-type convergence results for a wide variety of RBFs interpolants. Let $\phi(r)$ be an RBF whose Fourier transform $\hat{\phi}$ decays at least algebraically, meaning that

$$\|\hat{\phi}(\omega)\| \le C(1+\|\omega\|^2)^{-\tau}, \tag{33}$$

where C is a constant and $\tau > 0$ is a parameter that depends on the smoothness properties of ϕ. Then, we can use Theorem 11 in [16] to bound the error

$$\begin{aligned}\|\nabla_\Sigma f(\mathbf{x}) - \nabla_\Sigma s_f(\mathbf{x})\|_{L_\infty(\Sigma)} &= \|\nabla_\Sigma(f(\mathbf{x}) - s_f(\mathbf{x}))\|_{L_\infty(\Sigma)} \le C_1\|f - s_f\|_{W^1_\infty(\Sigma)}\\ &\le C_1 h^{\tau-5/2}\|f\|_{\mathcal{N}_\phi(\Sigma)} = \mathcal{O}(h^{\tau-5/2})\end{aligned} \tag{34}$$

for the surface gradient as in [17], and the error

$$\begin{aligned}\|\Delta_\Sigma f(\mathbf{x}) - \Delta_\Sigma s_f(\mathbf{x})\|_{L_\infty(\Sigma)} &= \|\Delta_\Sigma (f(\mathbf{x}) - s_f(\mathbf{x}))\|_{L_\infty(\Sigma)} \leq C_2 \|f - s_f\|_{W^2_\infty(\Sigma)} \\ &\leq C_2 h^{\tau - 7/2} \|f\|_{\mathcal{N}_\phi(\Sigma)} = \mathcal{O}(h^{\tau-7/2}) \end{aligned} \tag{35}$$

for the surface Laplacian. In these expressions,

$$h = \sup\{\min(\mathrm{d}_\Sigma(\mathbf{x}, \mathbf{x_i}))\}, \text{ with } \mathbf{x} \in \Sigma, \mathbf{x_i} \in \mathbf{X}, \tag{36}$$

is the the mesh norm, where $\mathrm{d}_\Sigma(\mathbf{x}, \mathbf{x_i})$ denotes the geodesic distance between $\mathbf{x}$ and $\mathbf{x_i}$ on Σ. In (34) and (35), C_1 and C_2 are constants, and $\|\cdot\|_{W^i_\infty(\Sigma)}$ and $\|\cdot\|_{\mathcal{N}_\phi(\Sigma)}$ are the Sobolev norm on Σ and the norm on the native space associated to the radial basis function $\phi(r)$ retricted to Σ, $\mathcal{N}_\phi(\Sigma)$, respectively.

Note that for the RBFs used in this paper (inverse multiquadrics) the Fourier transform $\hat{\phi}(\omega)$ decays exponentially and, hence, τ is greater than $(m + 7/2)$ for any $m > 0$. Therefore, according to (34) and (35) the convergence is spectral, i.e., it is faster than $\mathcal{O}(h^m)$ for any $m > 0$.

4 Numerical Experiments

4.1 Test Problem

To analyze the convergence properties of the method proposed in this paper (see Algorithm 1), we consider in Fig. 1 the direct problem

$$\Delta_{\mathbb{S}^2} f(\mathbf{x}) = g(\mathbf{x}) \tag{37}$$

to compute the surface Laplacian of the function $f(\mathbf{x}) = x\, y + z$ on the unit sphere $\mathbb{S}^2$, which is $g(\mathbf{x}) = -2(z + 3\, x\, y)$. In this case, the surface is explicitly known. A direct computation gives the normal vectors $\mathbf{n} = \mathbf{r}/r$, and the curvature $\kappa = 2$ at any point on the surface (see "Appendix A", where these expressions are derived). We choose the inverse multiquadric (IMQ) RBF $\phi(r) = 1/\sqrt{1 + \epsilon^2 r^2}$ with shape parameter ϵ for the computations.

Algorithm 1 Algorithm for computing $\Delta_{\mathbb{S}^2} f(\mathbf{x})$ in (37)

Require: Set of points $\{\mathbf{x}_i\}_{i=1}^N$ on $\mathbb{S}^2$, and vector $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]$.
Choose the radial basis functions $\phi(r)$.
Construct the interpolation matrix $A = [A]_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$.
Construct the matrix $C = [C]_{ij} = \Delta_\Sigma \phi(r_i(\mathbf{x}_j))$ using (19) (or (20)).
Compute the surface Laplacian matrix $L_\Sigma = C\, A^{-1}$.
Compute $L_\Sigma\, \mathbf{f}$ to find $\mathbf{g} = [g(\mathbf{x}_1), \ldots, g(\mathbf{x}_N)]$.

We consider a set of scattered points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ randomly placed on $\Sigma = \mathbb{S}^2$, as it is shown in Fig. 1d. We first evaluate how well the function $g(\mathbf{x})$ is represented by an RBF interpolant. In Fig. 1a, we display the interpolation error

$$E_I = \|s_g(\mathbf{x}_j) - g(\mathbf{x}_j)\|_\infty \text{ with } \mathbf{x}_j \in \mathbb{S}^2 \tag{38}$$

as function of the shape parameter ϵ for different number of points N on the surface of the sphere. To estimate (38) we have evaluated $\|s_g(\mathbf{x}_j) - g(\mathbf{x}_j)\|$ on another set of 4000 points $\{\mathbf{x}_j\}$ randomly distributed on the sphere. As expected, the interpolation error decreases as

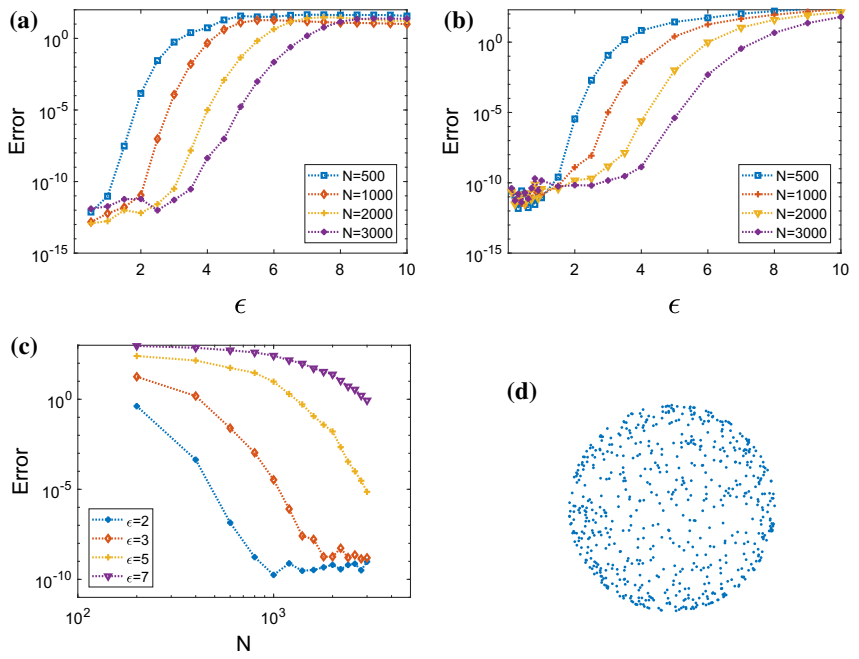

Fig. 1 Test problem (37): the surface is defined explicitly. **a** Interpolation error (38) as a function of the shape parameter ϵ, for different number of nodes N. **b** Error in the computation of the surface Laplacian (39) as function of the shape parameter ϵ for different number of nodes N. **c** Error (39) as function of N for different shape parameters ϵ. **d** Sphere with $N = 500$ randomly place nodes

ϵ diminishes, until certain value (not shown here) beyond which the matrix A becomes ill-conditioned and the interpolation errors cannot be further reduced. The results show the expected spectral convergence with respect to the shape parameter ϵ.

We note that there are different types of near-uniform node distributions with which the interpolation errors could be further reduced; see, for example, [12,14,39] for details about quasi-uniformly distributed maximal determinant node sets. However, we are interested in the case in which the surface is defined by a set of scattered points, so we limit the discussion to random point distributions.

It is apparent that for any fixed ϵ the error in the approximation of the surface Laplacian of $f(\mathbf{x})$ cannot be smaller than E_I. In Fig. 1b, we show the error

$$E_{\Delta_\Sigma} = \|L_\Sigma \mathbf{f} - \mathbf{g}\|_\infty \tag{39}$$

as function of ϵ for different number of points N on the surface of the sphere. We also observe the expected spectral convergence with respect to ϵ for all number of points N. Figure 1b shows errors of the same order of magnitude as those in Fig. 1a. Finally, in Fig. 1c we observe that the method converges spectrally in space as it was shown in Sect. 3.4.

We note that it is straightforward to solve the inverse problem associated to (37) if the function $g(\mathbf{x})$ is given and we seek the function $f(\mathbf{x})$. In that case, we just need to replace the last line in Algorithm 1 by the computation of $A\,C^{-1}\,\mathbf{g}$ to obtain $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]$.

So far, we have assumed that the surface over which we had to compute the surface Laplacian was known and, thus, the normal vector $\mathbf{n}$ and the curvature κ could be computed analytically; $\mathbf{n} = \mathbf{r}/r$ and $\kappa = 2$ in Fig. 1. In the next numerical experiments the surface is not explicitly known, but it is defined by a cloud of points. Therefore, the surface will be found by interpolation by solving problem (5), and the normal vectors and the curvature will be estimated using formulas (25)–(26) and (31), respectively; see Algorithm 2.

Algorithm 2 Algorithm for computing $\mathbf{n}$ and κ

Require: Set of points $\{\mathbf{x}_i\}_{i=1}^{N}$ on Σ.

Add the off-surface points $\{\mathbf{x}_i\}_{i=N+1}^{N+M}$.
Construct the vector $[\mathbf{f}]_i = [\Psi(\mathbf{x}_i)] = 0, 1, -1$ for $\mathbf{x}_i$ on, inside and outside of Σ.
Choose the radial basis functions $\phi(r)$.
Construct the interpolation matrix $A = [A]_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$, $i, j = 1, ..., N + M$.
Construct $\Psi(\mathbf{x})$ defined in (24) by finding the coefficients $\mathbf{d} = A^{-1}\mathbf{f}$.
Compute $\nabla\Psi(\mathbf{x}_i)$ and $\|\nabla\Psi(\mathbf{x}_i)\|$ from (26).
Compute $\mathbf{n}$ and κ using (25) and (31).

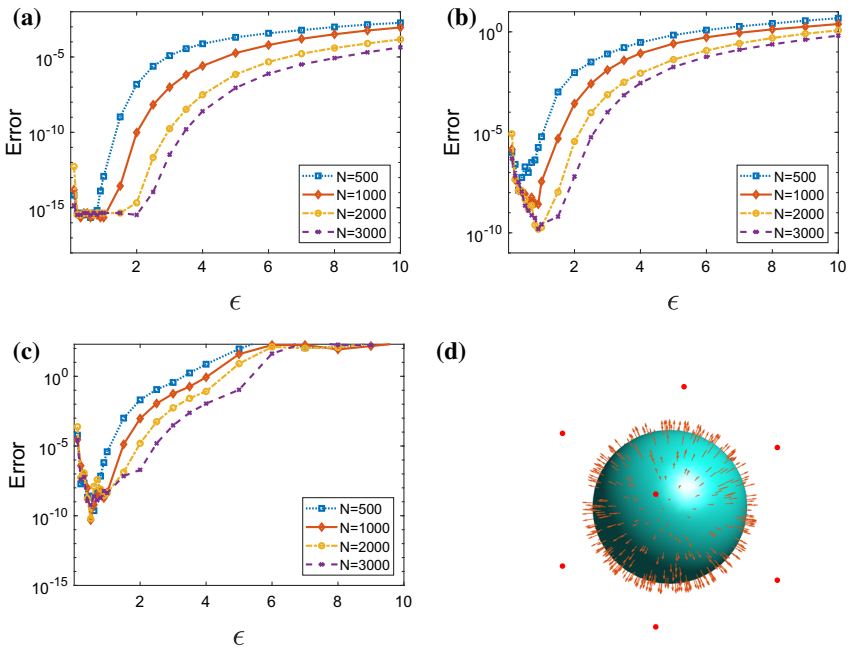

Fig. 2 Test problem (37): the surface is defined by a cloud of points. **a** Error (40) of the estimated normal vectors as function of ϵ for different number of nodes N. **b** Error (41) of the estimated curvature as function of ϵ for different number of nodes N. **c** Error (39) as function of N for different shape parameters ϵ. **d** Reconstruction of the sphere using the $N = 600$ surface points in Fig. 1. Seven of the $M = 9$ off-surface additional points used for the reconstruction are depicted in red. We also show the estimated normal vectors (Color figure online)

In Fig. 2d we display the reconstruction of the surface of the unit sphere found by solving (5) with the surface points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ shown in Fig. 1d. We also show the normal vectors obtained from (25)–(26). In this example, we have considered an IMQ with $\epsilon = 2$. In Fig. 2a, b, we show the error of the estimated normal vectors

$$E_{\mathbf{n}} = \|\cos(\mathbf{x}_i, \mathbf{n}_i) - 1\|_\infty, \quad \mathbf{x}_i \in \mathbf{X}, \tag{40}$$

and the error of the estimated curvature

$$E_\kappa = \|\kappa(\mathbf{x}_i) - 2\|_\infty, \quad \mathbf{x}_i \in \mathbf{X}, \tag{41}$$

as function of ϵ for different number of nodes N. We observe that both the normal vectors and the curvature are estimated with high precision for small values of the shape parameter ϵ. Still, the errors associated to the computation of the curvature are significant larger than those associated to the normal vectors. Finally, in Fig. 2c we plot the error in the Laplace–Beltrami operator computed with the approximate normals and curvature (compare it to the one using the exact expressions from Fig. 1b).

4.2 A Bioelectric Cardiac Model

We now apply the proposed method to a bioelectric cardiac source model proposed by Mitchell and Schaeffer [26], and used in [1] for shape reconstruction of cardiac ischemia. This is a simple two-current model that is able to reproduce the restitution properties of the cardiac tissue, as well as other complex electrophysiological behaviors such as spatial variations of the action potential duration and arrhythmogenic factors like alternans and discordant alternans. The model consists of two differential equations

$$\frac{dv}{dt} = \sigma \Delta_\Sigma v + J_{in}(v, h) + J_{out}(v) + J_{stim}(t), \tag{42}$$

and

$$\frac{dh}{dt} = G(h, v) = \begin{cases} \dfrac{1-h}{\tau_{open}}, & v < v_{crit} \\ \dfrac{-h}{\tau_{close}}, & v > v_{crit} \end{cases} \tag{43}$$

for the transmembrane voltage $v = v(\mathbf{x}, t)$, and the inactivation gate variable $h = h(\mathbf{x}, t)$. Both equations are written in dimensionless form. The variables v and h have units of time and they are scaled to vary between 0 and 1. In (42), the diffusive term represents the transmembrane current $i_m = \sigma \Delta_\Sigma v$ flowing through the cardiac cell membrane Σ,

$$J_{in}(v, h) = \frac{h(1-v)v^2}{\tau_{in}} \tag{44}$$

represents sodium and calcium currents going *inwards* the membrane Σ,

$$J_{out}(v) = -\frac{v}{\tau_{out}} \tag{45}$$

represents potassium currents going *outwards* the membrane, and J_{stim} is the external stimulus current. There are six parameters in the model: the conductivity of the cardiac tissue σ, the four phases of the cardiac action potential $\tau_{in}, \tau_{close}, \tau_{out}$, and τ_{open}, and the change-over voltage v_{crit}. In the simulations presented below, we set these parameters to $\sigma = 10^{-3}\,\mathrm{cm}^2/\mathrm{ms}$, $\tau_{in} = 0.2\,\mathrm{ms}$, $\tau_{close} = 150\,\mathrm{ms}$, $\tau_{out} = 10\,\mathrm{ms}$, $\tau_{open} = 130\,\mathrm{ms}$, and $v_{crit} = 0.13$. We refer the interested reader to [1,26] for more details about this model.

Equations (42)–(43) are supplemented with initial conditions

$$v(\mathbf{x}, t = 0) = 0 \quad \text{and} \quad h(\mathbf{x}, t = 0) = 1 \quad \forall\, \mathbf{x} \in \Sigma,$$

meaning that the membrane is initially at rest. Finally, to activate the current through the cardiac cell membrane we apply the stimulus current

$$J_{stim}(\mathbf{x}, t) = H(t_{stim} - t) e^{\frac{(\mathbf{x}-\mathbf{x}_s)^2}{\delta^2}}, \quad t \geq 0. \tag{46}$$

Here, $H(t)$ denotes the Heavisade function, t_{stim} the time when the stimulus ends, $\mathbf{x}_s$ the position where the stimulus is applied, and δ its spatial width.

4.2.1 Time Stepping

Given a set of sampled points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ on the surface Σ, we introduce the $N \times 1$ column vectors $\mathbf{v}_j$ and $\mathbf{h}_j$ whose i-th entries are the values of the transmembrane voltage v and the inactivation gate variable h at node $\mathbf{x}_i$ and time t_j, respectively. Using the method explained above, we approximate the diffusive term $i_m(\cdot, t_j) = \sigma \Delta_\Sigma \mathbf{v}_j$ in (42) by $i_m(\cdot, t_j) \approx \sigma L_\Sigma \mathbf{v}_j$ applying (22).

To integrate the equations in time we use the classical fourth order Runge–Kutta scheme. Thus, we define the column vectors

$$\mathbf{F}(\mathbf{v}_j, \mathbf{h}_j, t_j) = \sigma L_\Sigma \mathbf{v}_j + \mathbf{J}_{in}(\mathbf{v}_j, \mathbf{h}_j) - \mathbf{J}_{out}(\mathbf{v}_j) + \mathbf{J}_{stim}(t_j)$$

and $\mathbf{G}(\mathbf{v}_j, \mathbf{h}_j)$ that represent the values of the right hand sides of (42) and (43) at the set of sampled points $\mathbf{X}$, respectively, and we apply

$$\mathbf{v}_{j+1} = \mathbf{v}_j + \frac{\Delta t}{6}\left(\mathbf{K}_1^F + 2\mathbf{K}_2^F + 2\mathbf{K}_3^F + \mathbf{K}_4^F\right) \tag{47}$$

$$\mathbf{h}_{j+1} = \mathbf{h}_j + \frac{\Delta t}{6}\left(\mathbf{K}_1^G + 2\mathbf{K}_2^G + 2\mathbf{K}_3^G + \mathbf{K}_4^G\right) \tag{48}$$

at each time step, with

$$\begin{aligned}
\mathbf{K}_1^F &= \mathbf{F}\left(\mathbf{v}_j, \mathbf{h}_j, t_j\right), \quad \mathbf{K}_2^F = \mathbf{F}\left(\mathbf{v}_j + \frac{\Delta t}{2}\mathbf{K}_1^F, \mathbf{h}_j + \frac{\Delta t}{2}\mathbf{K}_1^G, t_j + \frac{\Delta t}{2}\right) \\
\mathbf{K}_3^F &= \mathbf{F}\left(\mathbf{v}_j + \frac{\Delta t}{2}\mathbf{K}_2^F, \mathbf{h}_j + \frac{\Delta t}{2}\mathbf{K}_2^G, t_j + \frac{\Delta t}{2}\right), \\
\mathbf{K}_4^F &= \mathbf{F}\left(\mathbf{v}_j + \Delta t\mathbf{K}_1^F, \mathbf{h}_j + \Delta t\mathbf{K}_1^G, t_j + \Delta t\right),
\end{aligned}$$

and similar expressions for $\mathbf{K}_1^G$, $\mathbf{K}_2^G$, $\mathbf{K}_3^G$, and $\mathbf{K}_4^G$.

4.2.2 Simulations on a 3-D Synthetic Cardioid

We first consider scattered nodes randomly distributed on the surface of a synthetic 3D heart muscle defined by

$$\Sigma = \left\{(x, y, z) \in \mathcal{R}^3 : \ (x - z^2)^2 + y^2 + z^2 - 1 = 0\right\}. \tag{49}$$

This cardioid models the surface of a heart muscle allowing us a detailed study of the method because both the normal vector and the curvature at any point of its surface are analytically known.

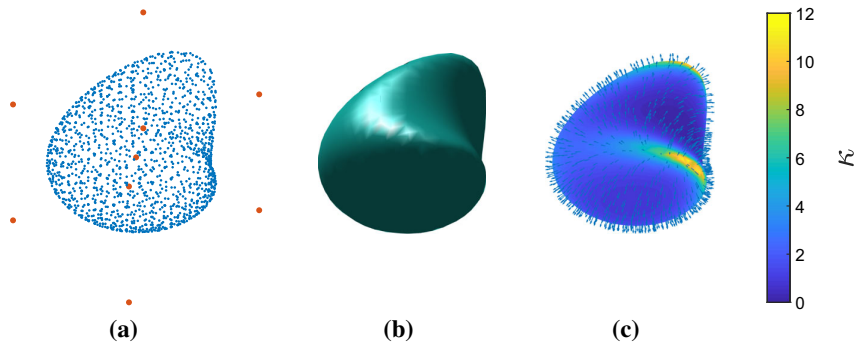

Fig. 3 3-D synthetic cardioid. **a** $N = 800$ nodes randomly place on its surface, and $M = 9$ off-surface nodes (red points). **b** RBF interpolant of the surface ($\Psi(\mathbf{x}) = 0$). **c** Normal vectors and color map of curvature term of the reconstructed surface (Color figure online)

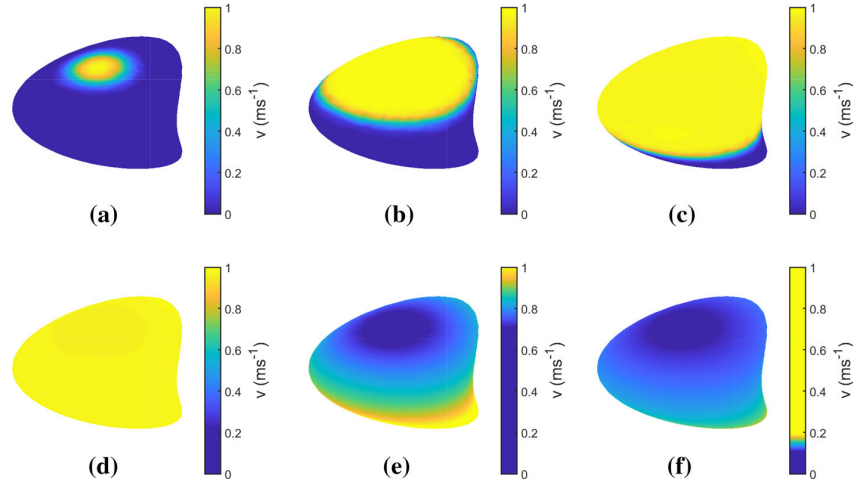

Fig. 4 Heartbeat cycle of 3-D synthetic cardioid. **a** Electric cardiac stimulus applied at $t = 0$. **b–f** Transmembrane voltages $v(\mathbf{x}, t)$ at $t = 40$, 100, 200, 300, and 400 ms, respectively

In Fig. 3a, we show the points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ that define the surface of the cardioid Σ, and the additional points off the surface needed for the interpolation of Σ. We have used $N = 800$ points randomly distributed on the surface, and $M = 9$ off-surface points (one inside and eight outside the surface). Using these points, and an IMQ with $\epsilon = 4$ as RBF, we first compute the coefficients d_i in (24) to represent the level set function $\Psi(\mathbf{x})$. In Fig. 3b we show the zero-level set of $\Psi(\mathbf{x})$ that represents the cardioid. Once a suitable representation of the surface of the heart muscle has been found, the unit normal vectors and the values of curvature are estimated using Algorithm 2; see Fig. 3c.

Finally, we approximate the surface Laplacian operator by the matrix L_Σ (23) using an IMQ with $\epsilon = 3$, and we integrate the model Eqs. (42)–(43) following the time stepping (47)–(48). Figure 4 displays a cycle of a heartbeat at different elapsed times. We show the transmembrane voltage $v(\mathbf{x}, t)$ on the surface of the heart, at $t = 0, 40, 100, 200, 300$, and

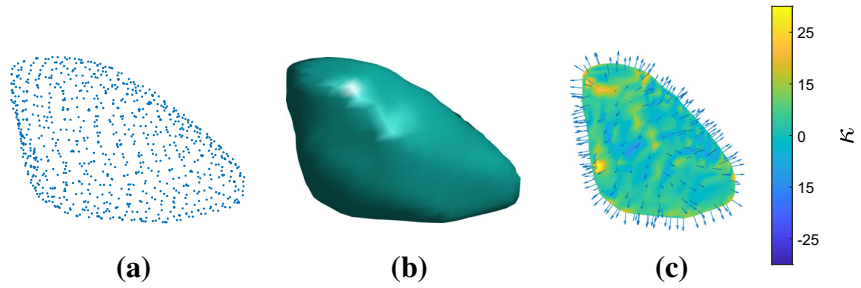

Fig. 5 Real heart surface obtained from a CT. **a** $N = 902$ nodes sampled on its surface. **b** RBF interpolant of the heart surface. **c** Normal vectors and curvature of the reconstructed surface

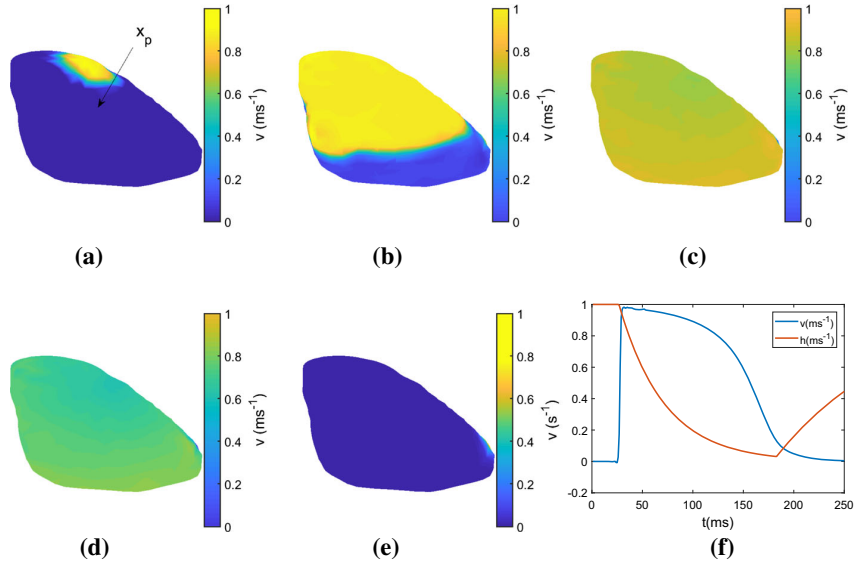

Fig. 6 Heartbeat cycle of realistic cardioid. **a** Electric cardiac stimulus applied at $t = 0$. **b–e** Transmembrane voltages $v(\mathbf{x}, t)$ at $t = 60$, 150, 300, and 400 ms, respectively. **f** Time dependence of transmembrane voltage $v(\mathbf{x}_p, t)$ and the h-gate variable $h(\mathbf{x}_p, t)$ at a point $\mathbf{x}_p$ on the surface

400 ms. At $t = 0$, the stimulus (46) with $t_{stim} = 0$ ms and $\delta = 0.2$ cm is applied at the upper part of the heart muscle (Fig. 4a). The stimulus is responsible for the electric impulse that initiates the cardiac cycle by creating an action potential. The cardiac tissue is then excited and, as a result, a cardiac electric wave propagates through the surface of the heart (Fig. 4b, c). For a while, all the cardiac tissue remains excited (Fig. 4d) and, finally, the heart muscle relaxes in preparation for the next stimulus (Fig. 4e, f).

4.2.3 Heart Surface from a CT

In the second numerical experiment we use data sampled on the surface of a realistic heart model. The surface nodes shown in Fig. 5a are obtained from a computerized tomography

(CT) of a 43 years old patient [8]. We depict the result of the RBF interpolation of the surface in Fig. 5b, and the normal vectors and the curvature of the surface in Fig. 5c. We have used $N = 980$ points on the surface of the heart, and exactly the same RBFs as in the previous numerical experiment.

Figure 6 displays the transmembrane voltage $v(\mathbf{x}, t)$ on the surface of the heart at $t = 0, 60, 150, 300$, and 400 ms. We also show the transmembrane voltage $v(\mathbf{x} = \mathbf{x}_p, t)$, and the h-gate variable $h(\mathbf{x} = \mathbf{x}_p, t)$, at a point $\mathbf{x}_p$ on the surface as function of time in Fig. 6f. The times at which the cardiac tissue experiences the different stages of a heartbeat correspond to those in [26], showing the success of the methodology proposed here for approximating the solution of this type of PDEs. We stress, again, that in these simulations the surface of the heart muscle is defined by the set of points where the data (the voltages) are measured. Neither the normal vectors nor the curvature at these points were assumed to be known.

5 Conclusions

In this paper, we have given an RBF-based closed formula for the Laplace–Beltrami operator which can be employed when the surface is defined by a cloud of points. The formula only depends on the positions of those points, the coefficients of the RBF surface reconstruction, and the first and second derivatives of the RBF at those points. If an explicit representation of the surface is available and, therefore, the normal vectors to the surface and its curvature can be computed analytically, the convergence analysis guarantees that the convergence of the formula is faster than algebraic. Our numerical experiments confirm these results. In the case where the surface is defined by a cloud of points, the accuracy of the approximations mainly depend on the errors made in the RBF surface reconstruction. As a realistic application that shows the usefulness of the proposed method, we have applied the formula for the Laplace–Beltrami operator for solving a PDE model for the electrical behavior on a cardiac surface.

Acknowledgements This work has been supported by Spanish MICINN Grant FIS2016-77892-R. We thank the anonymous reviewer for his or her careful reading of our manuscript and his or her many insightful comments and suggestions.

Appendix A: Laplace–Beltrami Operator of an RBF on the Unit Sphere

We show that the expression for the Laplace–Beltrami operator of an RBF defined on a general surface (19) reduces to (20), when the given nodes $\{\mathbf{x}_i\}$, $i = 1, \ldots, N$, are located on the unit sphere.

Let $\Psi(\mathbf{x}) = x^2 + y^2 + z^2 - 1$ be the function whose zero level set defines the unit sphere

$$\mathbb{S}^2 = \{\mathbf{x} \in \mathcal{R}^3 \ : \ \Psi(\mathbf{x}) = 0\} \tag{50}$$

implicitly. Then, $\nabla\Psi(\mathbf{x}) = 2\mathbf{x}$, so using (25) we obtain that $\mathbf{n}(\mathbf{x}) = \mathbf{x}$. Similarly, we find that $\Delta\Psi(\mathbf{x}) = 6$ and $\mathrm{D}_{\mathbf{n}}(\nabla\Psi(\mathbf{x})) = (\mathbf{x}\cdot\nabla)\,2\mathbf{x} = 2\mathbf{x}$, so the curvature is just $\kappa(\mathbf{x}) = 2$. Then, using (19) it follows that

$$\Delta_{\mathbb{S}^2}\phi(r_i(\mathbf{x})) = \left(1 + \frac{(\mathbf{x}\cdot\mathbf{r}_i)^2}{r_i^2} - 2\,\mathbf{x}\cdot\mathbf{r}_i\right)\frac{1}{r_i}\frac{\mathrm{d}\phi(r_i)}{\mathrm{d}r_i} + \left(1 - \frac{(\mathbf{x}\cdot\mathbf{r_i})^2}{r_i^2}\right)\frac{\mathrm{d}^2\phi(r_i)}{\mathrm{d}r_i^2}. \tag{51}$$

Since $\mathbf{x}_i, \mathbf{x} \in \mathbb{S}^2$, $\|\mathbf{x}_i\| = \|\mathbf{x}\| = 1$, so from the law of cosines $\|\mathbf{x}_i\|^2 = \|\mathbf{r}_i - \mathbf{x}\|^2 = r_i^2 + \|\mathbf{x}\|^2 - 2r_i\|\mathbf{x}\|\cos(\mathbf{r}_i, \mathbf{x})$ we find that $r_i = 2\cos(\mathbf{r}_i, \mathbf{x})$ and, hence, $\mathbf{x} \cdot \mathbf{r_i} = \|\mathbf{x}\| r_i \cos(\mathbf{r}_i, \mathbf{x}) = r_i^2/2$. Using this result in (51) we finally obtain (20).

References

1. Alvarez, D., Alonso-Atienza, F., Rojo-Alvarez, J.L., Garcia-Alberola, A., Moscoso, M.: Shape reconstruction of cardiac ischemia from non-contact intracardiac recordings: a model study. Math. Comput. Model. **55**, 1770–1781 (2012)
2. Ayton, G.S., McWhirter, J.L., McMurty, P., Voth, G.A.: Coupling field theory with continuum mechanics: a simulation of domain formation in giant unilamellar vesicles. Biophys. J. **88**, 3855–3869 (2005)
3. Barreira, R., Elliott, C.M., Madzvamuse, A.: The surface finite element method for pattern formation on evolving biological surfaces. J. Math. Biol. **63**, 1095–1119 (2011)
4. Bayona, V., Moscoso, M., Carretero, M., Kindelan, M.: RBF-FD formulas and convergence properties. J. Comput. Phys. **229**, 8281–8295 (2010)
5. Bertalmío, M., Bertozzi, A., Sapiro, G.: Navier–Stokes, fluid dynamics, and image and video inpainting. In: Proceedings of IEEE-CVPR, pp. 355–362 (2001)
6. Buhmann, M.: Radial Basis Functions. Cambridge University Press, Cambridge (2003)
7. 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 (2001). https://doi.org/10.1145/383259.383266
8. Chavez, C., Zemzemi, N., Coudiere, Y., Alonso-Atienza, F., Alvarez, D.: Inverse Problem of Electrocardiography: Estimating the Location of Cardiac Ischemia in a 3D Realistic Geometry. Lecture Notes in Computer Science, vol. 9126. Springer, Cham (2015)
9. Driscoll, T.A., Fornberg, B.: Interpolation in the limit of increasingly flat radial basis functions. Comput. Math. Appl. **43**, 413–422 (2002)
10. Dziuk, G., Elliott, C.M.: Finite element methods for surface PDEs. Acta Numer. **22**, 289–396 (2013)
11. Floater, M., Hormann, K.: Surface parameterization: a tutorial and survey. In: Dodgson, N.A., Floater, M.S., Sabin, M.A. (eds.) Advances in Multiresolution for Geometric Modelling, pp. 157–186. Springer, Berlin (2005)
12. Flyer, N., Wright, G.B.: Transport schemes on a sphere using radial basis functions. J. Comput. Phys. **226**, 1059–1084 (2007)
13. Flyer, N., Wright, G.B.: A radial basis function method for the shallow water equations on a sphere. Proc. R. Soc. A **465**, 1949–1976 (2009)
14. Fornberg, B., Piret, C.: A stable algorithm for flat radial basis functions on a sphere. SIAM J. Sci. Comput. **30**, 60–80 (2007)
15. Fonberg, B., Flyer, N.: A primer on radial basis functions with applications to geosciences. In: CBMS-NSF Regional Conference Series in Applied Mathematics (2015)
16. Fuselier, E.J., Wright, G.B.: Scattered data on embedded submanifolds with restricted positive definite kernel: Sobolev error estimates. SIAM J. Numer. Anal. **50**, 1753–1776 (2012)
17. Fuselier, E.J., Wright, G.B.: A high-order kernel method for diffusion and reaction–diffusion equations on surfaces. J. Sci. Comput. **56**, 535–565 (2013)
18. González-Rodríguez, P., Bayona, V.-, Moscoso, M., Kindelan, M.: Laurent series based RBF-FD method to avoid ill-conditioning. Eng. Anal. Bound. Elem. **52**, 24–31 (2015)
19. Greer, J.B.: An improvement of a recent Eulerian method for solving PDEs on general geometries. J. Sci. Comput. **29**, 321–352 (2006)
20. James, A.J., Lowengrub, J.: A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant. J. Comput. Phys. **201**, 685–722 (2004)
21. Kansa, E.J.: Multiquadrics, a scattered data approximation scheme with applications to computational fluid dynamics. I. Surface approximations and partial derivatives estimates. Comput. Math. Appl. **19**, 127–145 (1990)
22. Kansa, E.J.: Multiquadrics, a scattered data approximation scheme with applications to computational fluid dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations. Comput. Math. Appl. **19**, 147–161 (1990)
23. Kindelan, M., Moscoso, M., González-Rodríguez, P.: Radial basis function interpolation in the limit of increasingly flat basis functions. J. Comput. Phys. **307**, 225–242 (2016)

24. Macdonald, C.B., Ruuth, S.J.: The implicit closest point method for the numerical solution of partial differential equations on surfaces. SIAM J. Sci. Comput. **31**, 4330–4350 (2009)
25. Memoli, F., Sapiro, G., Thompson, P.: Implicit brain imaging. Hum. Brain Mapp. **23**, 179–188 (2004)
26. Mitchell, C.C., Shaeffer, D.G.: A two-current model for the dynamics of cardiac membrane. Bull. Math. Biol. **65**, 767–793 (2003)
27. Micchelli, C.A.: Interpolation of scattered data: distance matrices and conditionally positive definite functions. Constr. Approx. **2**, 11–22 (1986)
28. Myers, T.G., Charpin, J.P.F.: A mathematical model for atmospheric ice accretion and water flow on a cold surface. Int. J. Heat Mass Trans. **47**, 5483–5500 (2004)
29. Piret, C.: The orthogonal gradients method: a radial basis functions method for solving partial differential equations on arbitrary surfaces. J. Comput. Phys. **231**, 4662–4675 (2012)
30. Ruuth, S.J., Merriman, B.: A simple embedding method for solving partial differential equations on surfaces. J. Comput. Phys. **227**, 1943–1961 (2008)
31. Schaback, R.: Multivariate interpolation by polynomials and radial basis functions. Constr. Approx. **21**, 293–317 (2005)
32. Shankar, V., Wright, G.B., Kirby, R.M., Fogelson, A.L.: A radial basis function (RBF)-finite difference (FD) method for diffusion and reaction–diffusion equations on surfaces. J. Sci. Comput. **63**, 745–768 (2016)
33. Shu, C., Ding, H., Yeo, K.S.: Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier–Stokes equations. Comput. Methods Appl. Mech. Eng. **192**, 941–954 (2003)
34. Stam, J.: Flows on surfaces of arbitrary topology. ACM Trans. Graph. **22**, 724–731 (2003)
35. Tang, P., Qiu, F., Zhang, H., Yang, Y.: Phase separation patterns for diblock copolymers on spherical surfaces: a finite volume method. Phys. Rev. E **72**, 016710 (2005)
36. Tolstykh, A.I., Shirobokov, D.A.: On using radial basis functions in a "finite difference mode" with applications to elasticity problems. Comput. Mech. **33**, 68–79 (2003)
37. Wright, G.B.: Radial basis function interpolation: numerical and analytical developments. Ph.D. thesis, University of Colorado, Boulder (2003)
38. Wright, G.B., Flyer, N., Yuen, D.A.: A hybrid radial basis function-pseudospectral method for thermal convection in a 3D spherical shell. Geophys. Geochem. Geosyst. **11**, Q07003 (2010)
39. Womersley, R.S., Sloan, I.: How good can polynomial interpolation on the sphere be? Adv. Comput. Math. **14**, 195–226 (2001)