The Orthogonal Gradients Method: a Radial Basis Functions Method for Solving Partial Differential Equations on Arbitrary Surfaces

Cécile Piret *

Université Catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Avenue G. Lemaitre, 4 1348 Louvain-la-Neuve, Belgium

Abstract

Much work has been done on reconstructing arbitrary surfaces using the radial basis function (RBF) method, but one can hardly find any work done on the use of RBFs to solve partial differential equations (PDEs) on arbitrary surfaces. In this paper, we investigate methods to solve PDEs on arbitrary stationary surfaces embedded in $\mathcal{R}^3$ using the RBF method. We present three RBF-based methods that easily discretize surface differential operators. We take advantage of the meshfree character of RBFs, which give us a high accuracy and the flexibility to represent the most complex geometries in any dimension. Two out of the three methods, which we call the orthogonal gradients (OGr) methods are the result of our work and are hereby presented for the first time.

Key words: Radial basis functions, RBF, Closest point method, Implicit surfaces, Level set method, Orthogonal Gradients method, OGr method

1 Introduction

We consider solving PDEs on stationary two-dimensional arbitrary manifolds that are embedded in the three-dimensional space. Further, we study the confinement of differential operators to the manifolds when applied to functions that are defined in $\mathcal{R}^3$. Although a discretization of the operator in $\mathcal{R}^3$ is straightforward to obtain via RBFs, a restriction of this operator to the surface can be quite tricky to compute. In differential geometry, the Laplace-Beltrami

* Corresponding author.
Email address: cecile.piret@uclouvain.be (Cécile Piret).

operator can be defined on the tangential plane of any manifold embedded in a Euclidean space. However, for a general surface, it is non-trivial to derive the expressions for this operator.

In this paper, we will focus on two general methods, which we call the RBF OGr method and the RBF projection method. These methods were designed to discretize surface operators simply and efficiently. They make use of the RBF discretization of operators in $\mathcal{R}^3$ and of the RBF expansion describing the surface. We present two versions of the RBF OGr method and compare them to the existing RBF projection method [1,2]. Fast algorithms for the RBF OGr method, as well as extensions to a wider range of PDEs and to higher dimensions will be subjects of follow-up papers.

PDEs on surfaces emerge from applications in a wide range of fields such as image processing, computer graphics, mathematical physics, fluid dynamics, or biology. We review some of the work that has been done in solving PDEs on arbitrary surfaces. Early favored applications such as solving the reaction-diffusion (RD) differential equations for texture synthesis, also important in image processing or epidemiology, illustrate the range of methods for solving PDEs on arbitrary surfaces, as well as their evolution. Section 2 contains a short survey of the methods used for solving PDEs on arbitrary stationary surfaces. Section 3 contains an introduction to the RBF method and to its applications in solving PDEs, and in reconstructing surfaces. We will then introduce the RBF OGr method in two of its versions and review the Projection Method in Section 4. Section 5 will show some numerical results and we will give a conclusion in Section 6. Finally, details of implementation will be given in the Appendices.

We note that the surfaces in the figures were all rendered with the Matlab routine isosurface.

2 Methods for Solving PDEs on Arbitrary Surfaces

Parametrization techniques

Developed in the $80s$ and $90s$, one of the earliest and most common approaches consisted in solving the PDE on a plane and to map the solution as a parametric texture to the surface [3]. However, the unlikelihood to find a single parametric function to describe complex surfaces resulted in having to sew patches of texture together. In [4], Stam was able to solve fluid flow models on $2D$ surfaces. He solved the PDE in curvilinear coordinates on Catmull-Clark subdivision surfaces, a generalization to bi-cubic B-splines, which are naturally sewed together and which can represent any smooth arbitrary topology.

In [5], Lui et al proposed to conformally map the entire surface to a rectangle on which the PDE was solved using well-known and accurate techniques. In general, however, a parametrization is impractical to construct and may introduce artificial singularities (e.g. spherical coordinates introduce singularities at the poles). One can find a survey of parametrization techniques in [6].

Finite difference techniques on triangulated surfaces

Another set of common methods consisted in solving the PDE on the triangulated or polygonal surfaces directly, such as in [7]. Although the surface did not need to be parametrized globally, the equations were complicated to discretize on the polygonal grid. Furthermore, the geometric primitives such as the curvature or the surface normals, were difficult to compute [8].

Finite element methods

In [9], a finite element (FE) technique was presented for solving elliptic equations (the same authors later extend the range of PDEs to parabolic equations in [10]). In this paper, the Laplace-Beltrami operator was represented in terms of the tangential gradient, by projecting the space function gradient in $\mathcal{R}^3$ onto the surface tangent plane via the formula $\Delta_S = \nabla_S.\nabla_S$ where $\nabla_S = (I - \vec{n}.\vec{n}^T)\nabla$. The surface was represented in terms of splines, rather than being globally parametrized. A similar technique was recently proposed in [11] to solve the Cahn-Hilliard equation.

Finite differences on $\mathcal{R}^3$ cartesian grid

In [8,12], the approach consisted in having an implicit representation of the surface via level sets, and a surface Laplacian representation in terms of tangential gradients. Furthermore, the extension of the function to nodes outside of the surface was defined in such a way that the gradients of the level set and of the function are orthogonal. This greatly simplified the finite difference formulas, and the embedding into $\mathcal{R}^3$ allowed computations to be performed on the fixed $\mathcal{R}^3$ cartesian grid, in a narrow band surrounding the surface [13]. However, the technique led to problems when solving diffusion equations. The projected operator which is degenerate in the direction normal to the surface, and the boundary conditions set in the band surrounding the surface can lead to inconsistencies. In view of this issue, the method is improved in [14,13]. Some limitations persist and can be found in [15].

The Closest Point method

The Closest Point Method was introduced in 2006 by Ruuth and Merriman [15]. It too consisted in embedding the surface operator into $\mathcal{R}^3$. The biggest differences with the methods mentioned above are that the method was based on a closest point function representation (versus a level set representation) and only the cartesian differential operators were being discretized (versus projected differential operators). The method can handle open surfaces and filamentary objects and the limitation to narrow bands around the surface of the computations does not impact the convergence of the algorithm. This technique has successfully been applied to a wide range of problems ([15–19]).

The Radial Basis Function Projection method

Most recently, Fuselier and Wright in [2] used the representation of the surface Laplacian in terms of tangential gradients in an RBF setting. The method consists in projecting the gradient onto the plane tangent to the surface, with the same projection operator as the techniques mentioned above. The method, which was previously introduced in the specific case of the sphere in [1], is extended to arbitrary surfaces.

The Radial Basis Function Orthogonal Gradients method

The orthogonal gradients method which we introduce here is also based on an RBF representation. The surface is implicitly defined as the RBF zero-isosurface of a 'distance' function defined in $\mathcal{R}^3$. Instead of using the gradient projection technique mentioned above, we adopt a closest point representation for the expansion approximating the solution. As we will see, the gradients of the distance function and of the PDE solution are set to be normal to each other at each point of the surface, which allows us to recover the surface restricted operators from the standard $\mathcal{R}^3$ discretized operators.

3 RBF methodology

3.1 The form of an RBF interpolant

The basic RBF interpolant takes the form

$$s(\vec{x}) = \sum_{i=1}^{N} \lambda_i \, \phi(\|\vec{x} - \vec{x}_i\|), \tag{1}$$

Name of RBF	Abbreviation	Definition
Smooth, global		
Multiquadric	MQ	$\sqrt{1+(\varepsilon r)^2}$
Inverse multiquadric	IMQ	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$
Inverse quadratic	IQ	$\frac{1}{1+(\varepsilon r)^2}$
Gaussian	GA	$e^{-(\varepsilon r)^2}$
Piecewise smooth, global		
Cubic	CU	$\lvert r\rvert^3$
Thin plate spline	TPS	$r^2 \ln \lvert r\rvert$

Table 1
Definitions of some types of radial functions. The shape parameter ε controls their 'flatness'.

where $||\cdot||$ denotes the Euclidean norm. In order for it to take the values f_i at locations $\vec{x}_i$, $i = 1, 2, \ldots n$, the expansion coefficients λ_i need to satisfy

$$A\,\vec{\lambda} = \vec{f}, \tag{2}$$

where the entries of the matrix A are $A_{i,j} = \phi(||\vec{x}_i - \vec{x}_j||)$. We denote numerical use of (2) followed by (1) as 'RBF-Direct'. In this study, we will concentrate our attention on the radial functions $\phi(r)$ listed in Table 1. The parameter ε, included in all but the piecewise smooth global cases CU and TPS, is known as the *shape parameter*.

3.2 Surface reconstruction

Rolland Hardy introduced RBFs in 1971, [20]. He originally presented the method for the multiquadric (MQ) radial function. In 1982, Richard Franke popularized the MQ method with his report on 32 of the most commonly used interpolation methods [21]. He subjected those methods to thorough tests, and found the MQ method overall to be the best one. Franke also conjectured the unconditional non-singularity of the interpolation matrix associated with the MQ radial function, but it was not until a few years later, in 1986, that Charles Micchelli [22] was able to prove it, making use of work by Schoenberg from the $30s$ and $40s$. The main feature of the MQ method is that the interpolant is a linear combination of translations of a basis function which only depends on the Euclidean distance from its center. This basis function is therefore radially symmetric with respect to its center. The MQ method was generalized to other

radial functions, such as the thin plate spline, the gaussian, the cubic, etc. and the method was called the Radial Basis Function method.

Implicit surfaces are difficult to specify [23]. The idea of implicitly representing a surface as the zero-isosurface of a $3D$ function with distance dependent functions has been developed in 1992 [24], then in Savchenko in 1995 [25] and in Turk and O'Brien [26] in 1999. It is not until 1999 with the work of Turk and O'Brien [27],[28], that the implicit representation of surfaces via global radial functions started to show great results (variational implicit functions). They formulated the surface reconstruction as a variational problem and they added off-surface points, where an energy functional had to be minimized, in order to reduce the unphysical oscillations. They realized that certain types of RBFs solve the minimization problem inherently (the thin plate spline in $2D$ and the biharmonic radial function in $3D$). However, a big problem in the RBF approach is its large computational complexity. The problem has been tackled in several papers and has led to fast RBF-based surface reconstruction methods. Among them, [29–31] are based on compactly supported RBFs, and [32], [33] use C_∞ radial functions with a fast multipole method (FMM), and make it possible to reliably and cheaply reconstruct surfaces of hundreds of thousand of nodes from a $3D$ node cloud.

In order to apply an operator to a function defined on a surface, it is important to have an accurate representation of this surface. Starting from a point cloud in $3D$, one can build an RBF expansion using a variational approach, defining the surface as a level-surface. Geometric characteristics of the surface such as normals, curvatures, etc. are thereby readily available. We use RBFs to find a function whose zero-level surface will approximate the point cloud's underlying manifold.

A standard technique to represent the surface as the iso-surface of an RBF expansion [34,32] is to append to the N-node point cloud, $2N$ nodes at a δ distance in the normal direction to the surface (N nodes inside and N nodes outside). We note that the parameter δ is important for the quality of the interpolant. One will associate the value of 0 to the original point cloud, -1 to the points inside the manifold and $+1$ to the points outside.

Although one can interpolate a constant function $f(\vec{x}) = c$ over the surface only, and suffer the already large complexity associated with N and not $3N$, the community seems to prefer appending extra nodes for the surface reconstruction. Indeed, adding layers adds a fair amount of stability. In the absence of extra layers, unnatural foldings of the 0-isosurface are frequent in the areas where the nodes are sparse. These foldings do not appear when we introduce the extra layers. Thankfully, several techniques have been introduced to deal with the huge complexity resulting from interpolating on $3N$ nodes (e.g. the Fast Multipole Method in [32], Partition of Unity Method in [35]).

As in Figure 1, we define the distance function

$$s(\vec{x}) = \sum_{i=1}^{N} \lambda_i \, \phi(\|\vec{x} - \vec{x}_i\|) + \mu_i \, \phi(\|\vec{x} - \vec{x}_i + \delta\vec{n}_{\vec{x}_i}\|) + \nu_i \, \phi(\|\vec{x} - \vec{x}_i - \delta\vec{n}_{\vec{x}_i}\|)$$

where $s(\vec{x}_i) = 0$, $s(\vec{x}_i + \delta\vec{n}_{\vec{x}_i}) = 1$ and $s(\vec{x}_i - \delta\vec{n}_{\vec{x}_i}) = -1$. The surface Γ is the zero-isosurface of $s(\vec{x})$.

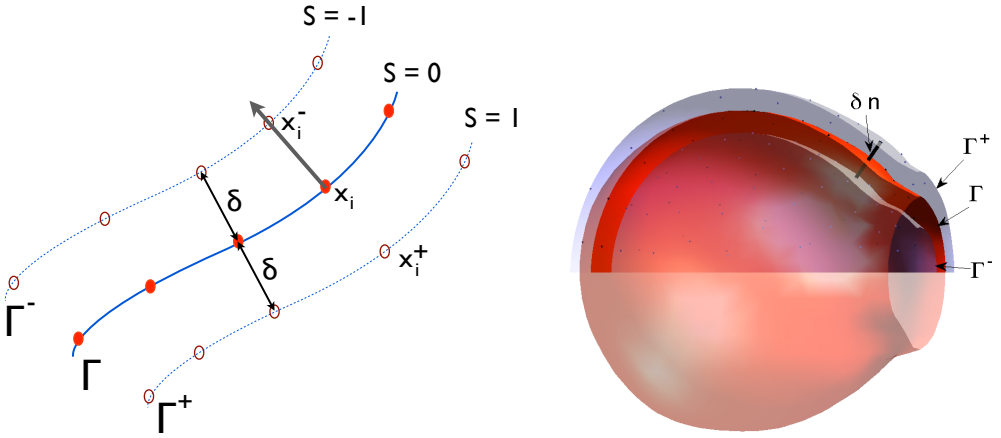

Fig. 1. Illustration of the surface reconstruction via RBFs. N points are uniformly distributed on the main surface Γ (in red). At each point, a rough approximation of the normal $\vec{n}$ to the surface is computed and two new points are obtained at distance δ from the surface, one on either side of it. The distance function is approximated by an RBF expansion which uses all $3N$ points. The 0-level surface of the distance function approximates the surface Γ and it is surrounded by the 1 and the -1-level surfaces, Γ^+ and Γ^- respectively.

3.3 RBFs for PDEs

It is in the 90s that Ed Kansa presented the RBF method as a discretization technique in the context of solving partial differential equations [36,37]. The RBF collocation approach in space consists in solving time dependent PDEs using the method of lines (MOL). The spatial differential operators are discretized via RBFs, by developing differentiation matrices, and the resulting system of ODEs is solved in time using a standard ODE solver. We will use this technique all along this paper. See the following references for instances where this technique is used to solve PDEs, in both cases on the surface of the sphere [1,38].

Computation of a Global Differentiation Matrix

We wish to find a matrix D that discretizes, via an RBF representation, the continuous differential operator L. Assuming that the solution takes the func-

tion values f on Γ, we require that

$$f(\vec{x}_i) = \sum_{j=1}^{N} \lambda_j \ \phi(\|\vec{x}_i - \vec{x}_j\|), \tag{3}$$

for all x_i, and it leads to the matrix equation $A\vec{\lambda} = f$. Analytically applying the differential operator to the radial function gives

$$g(\vec{x}_i) = \sum_{j=1}^{N} \lambda_j \ L\phi(\|\vec{x}_i - \vec{x}_j\|), \tag{4}$$

where $g(\vec{x}_i)$ is the value of the underlying function's derivative at each x_i. Thus $B\vec{\lambda} = g$ in matrix form, where $B_{i,j} = L\phi(||\vec{x} - \vec{x}_j||)_{x=x_i}$. The collocation matrix A is unconditionally nonsingular. This allows us to eliminate the expansion coefficient vector $\vec{\lambda}$ leading to $g = BA^{-1}f$. The differentiation matrix $D = BA^{-1}$ gives an RBF discretization of L.

4 Methods for constructing surface operators

Once δ has been fixed, and that the distance function $s(\vec{x})$ has been defined as illustrated in Figure 1, surface differential operators can be computed.

One can expand ∇f in any orthonormal coordinate system. Let $\{\vec{n}, t\vec{1}, t\vec{2}\}$ be the normal, first tangent and second tangent directions respectively, taken at some point $\vec{x} \in \Gamma$.

$$\nabla f = \partial_n f \ \vec{n} + \partial_{t_1} f \ \vec{t_1} + \partial_{t_2} f \ \vec{t_2}$$

The surface gradient, ∇_Γ,is the projection of ∇ on the plane tangent to the Γ at point $\vec{x} \in \Gamma$. Thus $\nabla_\Gamma = \vec{t_1} \ \partial_{t_1} + \vec{t_2} \ \partial_{t_2}$. We consider two approaches to extract $\nabla_\Gamma f$ out of the computation of ∇f:

- *Projection.* $\nabla_\Gamma = \nabla f - \partial_n f \vec{n}$. Thus, $\nabla_\Gamma = (I - \vec{n}\vec{n}^T)\nabla f$. This technique, used for discretizing operators restricted to arbitrary surfaces, can be commonly found in the literature applied to finite differences and finite element methods [9,10,8,12,14,13], and was most recently introduced in the context of the radial basis functions method, [2].

- *Orthogonal gradients.* We extend f outside of Γ, and impose that the normal component of its gradient is nul. We thus require that ∇f be orthogonal to ∇s, the gradient of the distance function, whose zero-level surface corresponds to Γ. We find this technique in the literature to simplify finite

difference formulas in [8], and as the core of the Closest Point Method [15]. We now introduce the latter approach in the context of RBFs, and refer to it as the RBF Orthogonal Gradients method.

4.1 The Orthogonal Gradients method

Consider a function f defined on an arbitrary surface Γ. We have an implicit level surface RBF representation for Γ, as in Figure 1. We define the Laplacian of f on Γ as follows

$$\Delta f = (\vec{n}\partial_n + \vec{t_1}\partial_{t_1} + \vec{t_2}\partial_{t_2}).(\vec{n}\partial_n + \vec{t_1}\partial_{t_1} + \vec{t_2}\partial_{t_2})f, \tag{5}$$

which will be also the surface Laplacian if both $\partial_n f$ and $\partial_n^2 f$ vanish everywhere. We will use the $2N$ extra nodes that were introduced in the surface reconstruction, to add $2N$ restrictions on f, and impose that both partials of f in the normal direction vanish. This is to say that physically, at each point on Γ, f will be constant in the direction normal to the surface. The low order version is next introduced, mostly with the purpose of better explaining the high order version. In practice, we will rather use the high order version as it gives much more accurate results.

4.1.1 Low order version

The approach adopted in the Closest Point method consists in imposing that the function values outside of Γ, be $f(\vec{x}) = f(cp(\vec{x}))$, where cp is the 'closest point' operator and maps $\vec{x}$ to its closest point on Γ. This insures that, to a certain algebraic order, $\partial_n^2 f$ will vanish. This is also what we do in an RBF setting, and the positioning of our nodes is perfectly suited for it.

Figure 1 shows the N nodes $\vec{x}_i$ located on Γ, surrounded with $2N$ extra nodes, $\vec{x}_i^{\pm}$, at a small distance δ along a rough estimate of the normal direction. Taking the $\mathcal{R}^3$ Laplacian of f while setting $f(\vec{x}^{\pm}) = f(\vec{x})$ gives us a rough estimate of the surface Laplacian of f.

We can compute the normals much more accurately. We will call the *true* normal direction, the direction that is normal to the RBF approximation of the surface. Note that it will be important from now on to use the true normal direction rather than rough estimates, because the OGr method relies on the perfect orthogonality between ∇f and ∇s. This true normal direction, which is thus associated with the zero level set of the distance RBF expansion $s(\vec{x})$, is readily available at each point $\vec{x}_i$: $\vec{n} = \frac{1}{\sqrt{s_x^2+s_y^2+s_z^2}}(s_x, s_y, s_z)$. One can find nodes $\vec{y}_i^{\pm}$ along the true normals, located at a small distance γ from $\vec{x}_i$ as shown in

Figure 2. Taking the $\mathcal{R}^3$ laplacian of f while imposing that $f(\vec{y}^{\pm}) = f(\vec{x})$ thus gives us a better estimate of the surface laplacian, since at $\vec{x} = \vec{x}_i$,

$$\begin{cases} \partial_n f(\vec{x}) \approx \frac{f(\vec{x}) - f(\vec{y}_i^{\pm})}{\gamma} = 0 \\ \partial_n^2 f(\vec{x}) \approx \frac{2f(\vec{x}) - f(\vec{y}_i{}^{+}) - f(\vec{y}_i{}^{-})}{\gamma^2} = 0 \end{cases}$$

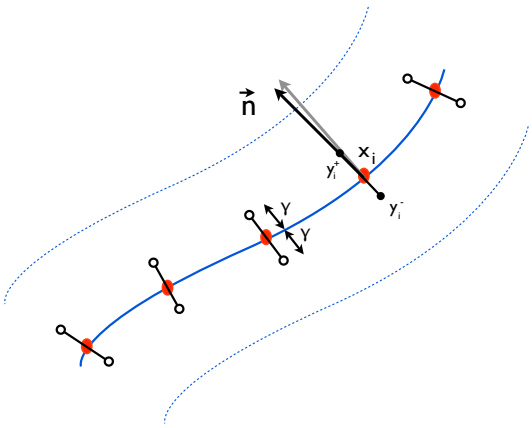

Fig. 2. Illustration of the RBF Orthogonal Gradients method. When the distance function has been defined using the $3N$ nodes, the true normals (to the approximation of the surface) $\vec{n}$ can be obtained at each point $\vec{x}_i$. We define new nodes $\vec{y}_i^{+}$ and $\vec{y}_i^{-}$ at a distance of γ from $\vec{x}_i$ on either side of Γ, along the true normal direction.

Implementation First, notice that both $f(\vec{x})$ and the level set distance function $s(\vec{x})$ have an RBF expansion defined on exactly the same centers. Let the function $f(\vec{x})$ be defined as follows

$$f(\vec{x}) = \sum_{i=1}^{N} \mu_i \, \phi(\|\vec{x} - \vec{x}_i\|) + \mu_i^{+} \, \phi(\left\|\vec{x} - \vec{x}_i^{+}\right\|) + \mu_i^{-} \, \phi(\left\|\vec{x} - \vec{x}_i^{-}\right\|)$$

We follow the standard procedure laid out in Section 2.2 to find the differentiation matrix D.

$$D = \begin{pmatrix} \Delta\Phi_x(x) & \Delta\Phi_{x^+}(x) & \Delta\Phi_{x^-}(x) \\ \Delta\Phi_x(x^+) & \Delta\Phi_{x^+}(x^+) & \Delta\Phi_{x^-}(x^+) \\ \Delta\Phi_x(x^-) & \Delta\Phi_{x^+}(x^-) & \Delta\Phi_{x^-}(x^-) \end{pmatrix} \begin{pmatrix} \Phi_x(x) & \Phi_{x^+}(x) & \Phi_{x^-}(x) \\ \Phi_x(y^+) & \Phi_{x^+}(y^+) & \Phi_{x^-}(y^+) \\ \Phi_x(y^-) & \Phi_{x^+}(y^-) & \Phi_{x^-}(y^-) \end{pmatrix}^{-1}$$

where $x^{\pm}$ belong to the layers $\Gamma^{\pm}$ as in Figure 1, where $y^{\pm}$ are the nodes illustrated in Figure 2, and where $\Phi_v(w)$ is a $N \times N$ matrix whose entries are RBF radial functions centered at points $\vec{v}$ and evaluated at points $\vec{w}$.

When applied to a function values vector, this matrix produces a vector containing the approximated surface Laplacian of the underlying function f, evaluated at each node in Γ, Γ^+, and Γ^-.

$$\begin{pmatrix} D_{1,1} & D_{1,2} & D_{1,3} \\ D_{2,1} & D_{2,2} & D_{2,3} \\ D_{3,1} & D_{3,2} & D_{3,3} \end{pmatrix} \begin{pmatrix} f_\Gamma \\ f_\Gamma \\ f_\Gamma \end{pmatrix} = \begin{pmatrix} g_\Gamma \\ g_{\Gamma^+} \\ g_{\Gamma^-} \end{pmatrix}$$

Since we are only interested in the values of function g_Γ, the surface Laplacian of $f(\vec{x})$ on Γ, and since the function values f are identical on all three surface layers Γ, Γ^+ and Γ^-, we can find the smaller sized $(N \times N)$ differentiation matrix $D = D_{1,1}+D_{1,2}+D_{1,3}$, such that $Df_\Gamma = g_\Gamma$. We note that, in the context of solving a time-dependent PDE, the inversion of the $3N \times 3N$ matrix can be performed in the preprocessing stage. At each time step, we will apply the $N \times N$ (and no longer the $3N \times 3N$) differentiation matrix, still leading to an operations count of $O(N^2)$, but with a much smaller constant.

4.1.2 High Order Version

One can further improve the above algorithm by directly imposing that $\partial_n f = 0$ and that $\partial_n^2 f = 0$. After noting that $\vec{n} \cdot \nabla f = \partial_n f$ and that $\partial_n^2 f = (\vec{n} \cdot \nabla)(\vec{n} \cdot \nabla f)$, we require that $\vec{n} \cdot \nabla f = 0$ and that $(\vec{n} \cdot \nabla)(\vec{n} \cdot \nabla) f = 0$, which is equivalent to the algorithm above in the limit of $\gamma \to 0$. Figure 3 shows the error in taking the surface Laplacian of $Y_{15}^3(x, y, z)$ on a unit sphere, via the low order and the high order methods. We notice that as $\gamma \to 0$, the error induced by the low order method converges to the error induced by the high order one. Figure 4 illustrates the core of the method and gives a justification for its name: the 'Orthogonal Gradients method'.

Implementation Once again, we will use the same centers as for the construction of the surface. We wish to impose the following

$$\begin{pmatrix} \Phi(\vec{x}) \\ \vec{n} \cdot \nabla \Phi(\vec{x})_{\vec{x}=\vec{x_i}} \\ (\vec{n} \cdot \nabla)(\vec{n} \cdot \nabla)\Phi(\vec{x})_{\vec{x}=\vec{x_i}} \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{0} \\ \vec{0} \end{pmatrix}$$

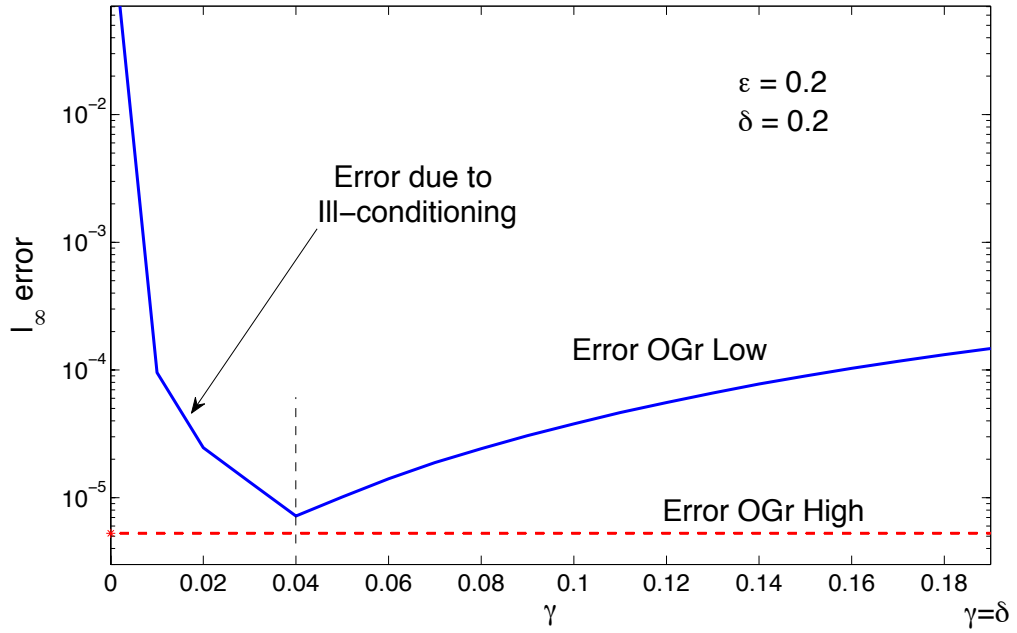

Fig. 3. Error in taking the surface Laplacian on the surface of the unit sphere of the spherical harmonic $Y_{15}^3(x, y, z)$, using the MQ RBF and 900 maximum determinant near uniformly distributed nodes. We show the error, computed via the low order OGr in blue and the high order OGr in red, versus the low order OGr parameter γ. As $\gamma \to 0$, the lower order method would be equivalent to the higher order one, if it weren't for the ill-conditioning it encounters in the small γ range.

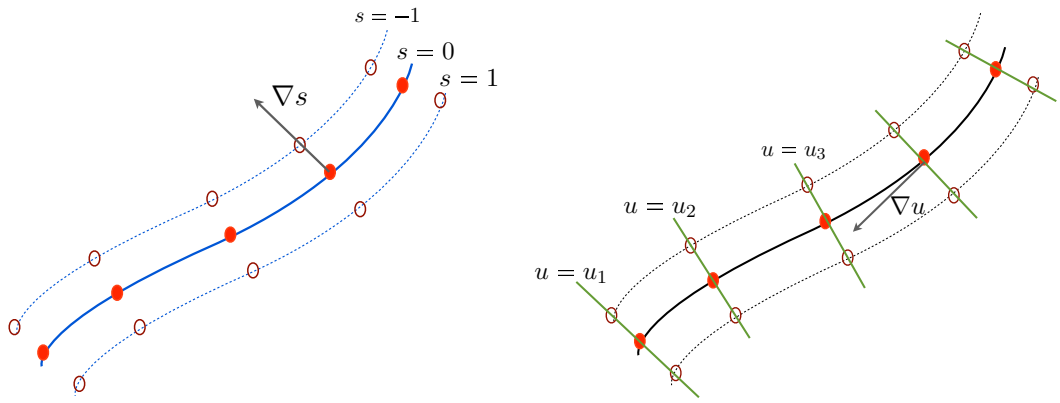

Fig. 4. Schema of the 'Orthogonal Gradients method'. On the left, the level-set distance function $s(\vec{x})$ is constant over Γ. This implies that, at each node on Γ, its gradient points in the direction normal to the surface at that point. Now consider function $u(\vec{x})$ on the right. Heuristically, by requiring that function $u(\vec{x})$ be constant in the direction normal to the surface (∇s), we impose that the gradients of functions $s(\vec{x})$ and $u(\vec{x})$ be orthogonal, and thus that the normal component of derivatives of $u(\vec{x})$ be null.

The differentiation matrix thus takes the form

$$D = \begin{pmatrix} \Delta\Phi_x(x) & \Delta\Phi_{x^+}(x) & \Delta\Phi_{x^-}(x) \\ \Delta\Phi_x(x^+) & \Delta\Phi_{x^+}(x^+) & \Delta\Phi_{x^-}(x^+) \\ \Delta\Phi_x(x^-) & \Delta\Phi_{x^+}(x^-) & \Delta\Phi_{x^-}(x^-) \end{pmatrix} \begin{pmatrix} \Phi_x(x) \\ \vec{n}\cdot\nabla\Phi(\vec{x})_{\vec{x}=\vec{x_i}} \\ (\vec{n}\cdot\nabla)(\vec{n}\cdot\nabla)\Phi(\vec{x})_{\vec{x}=\vec{x_i}} \end{pmatrix}^{-1}$$

and

$$\begin{pmatrix} D_{1,1} & D_{1,2} & D_{1,3} \\ D_{2,1} & D_{2,2} & D_{2,3} \\ D_{3,1} & D_{3,2} & D_{3,3} \end{pmatrix} \begin{pmatrix} f_\Gamma \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} g_\Gamma \\ g_{\Gamma^+} \\ g_{\Gamma^-} \end{pmatrix}$$

which reduces to $D_{1,1} f_\Gamma = g_\Gamma$. One can thus instead use the $N \times N$ differentiation matrix $D = D_{1,1}$. We note like in Section 4.1.1, that in the context of solving a time-dependent problem, although the operations count for computing the $3N \times 3N$ differentiation matrix is of $O(N^3)$ with a large constant, it can be performed in the preprocessing stage. Each time step will *only* involve the application of a $N \times N$ (and not a $3N \times 3N$) matrix. The operations count will thus be still of $O(N^2)$, but with a much smaller constant. Another form of the differentiation matrix is given in Appendix 1, while Appendix 2 gives details on how to compute $\vec{n}\cdot\nabla\Phi(\vec{x})_{\vec{x}=\vec{x_i}}$ and $(\vec{n}\cdot\nabla)(\vec{n}\cdot\nabla)\Phi(\vec{x})_{\vec{x}=\vec{x_i}}$.

4.2 The Operator Projection Method on Surfaces

For the sake of comparison, we give a brief overview of the method of projection, which was introduced in the context of RBFs in [1,2]. It consists, for each point on the surface, in projecting the gradient of the function defined on the surface onto the plane that is tangent to the surface, by applying $(I - \vec{n}\vec{n}^T)$ to the operator.

$$\begin{aligned} (I - \vec{n}\vec{n}^T)\nabla f &= (I - \vec{n}\vec{n}^T)(\partial_n f\ \vec{n} + \partial_{t1} f\ \vec{t1} + \partial_{t2} f\ \vec{t2}) \\ &= \partial_{t1} f\ \vec{t1} + \partial_{t2} f\ \vec{t2} \\ &= \nabla_\Gamma f \end{aligned}$$

Implementation We define the projection operator at a point $\vec{x}_i$ as $P_{t,\vec{x}_i} = I - \vec{n}_i\vec{n}_i^T$, where $\vec{n}_i$ is the unit vector that is normal to the surface at node $\vec{x}_i$. We can thus represent the differential operators restricted to the surface Γ, as

a linear combination of differential operators in $3D$.

$$\begin{pmatrix} \partial_{\Gamma,x} \\ \partial_{\Gamma,y} \\ \partial_{\Gamma,z} \end{pmatrix}_{\vec{x}=\vec{x_i}} = \begin{pmatrix} \\ I - \vec{n}_i \vec{n}_i^T \\ \\ \end{pmatrix} \begin{pmatrix} \partial_x \\ \partial_y \\ \partial_z \end{pmatrix}_{\vec{x}=\vec{x}_i}$$

Thus, the differentiation matrix approximating the surface restricted partial derivative in the x-direction $\partial_{\Gamma,x}$ can be written as $D_{\Gamma,x} = C_1 D_x + C_2 D_y + C_3 D_z$ where $C_1 = diag(1 - n_x(1)^2)$, $C_2 = diag(n_x(1)n_x(2))$, $C_3 = diag(n_x(1)n_x(3))$ and where D_x, D_y and D_z are the differentiation matrices approximating ∂_x, ∂_y and ∂_z respectively. The differentiation matrices $D_{\Gamma,y}$ and $D_{\Gamma,z}$ are similarly computed.

4.3 Fundamental Difference Between the Techniques

The OGr and projection methodologies are fundamentally different. While the OGr method relies on modifying the RBF expansion of the solution, by adding requirements on its derivatives, the projection method modifies the operator only. Both techniques have been applied to different methods (FEM, FD,CPM). In the contexts of FEM and FD methods, the technique of projection has lead to problems in solving diffusion problems. Indeed, the operator to which it leads is degenerate in the direction normal to the surface [39]. Difficulties have been encountered while trying to solve as simple problems as the heat equation[14]. Furthermore, the coupling of these two techniques [8,12] has also led, in certain circumstances, to large errors. It is actually the subject of [14,13]. In the latter, a different projection operator is proposed, which still does not produce optimal results [15].

4.4 Parameters

In the above methods, the parameters need to be properly defined to guarantee a good accuracy. These are the shape parameter ϵ and the offset parameter δ. Both of these parameters have a direct impact on both the accuracy and the conditioning of the differentiation matrix. A lot of work has been done on the search for an *optimal* shape parameter ϵ. Since the topology and the nodes' distribution and density also impact the conditioning, finding a formula for this optimal' ϵ is near impossible. As ϵ gets smaller, the accuracy improves but the conditioning worsens [34,40]. Unless one uses one of the algorithms that bypass the small shape parameter conditioning issue [41,42,38,43], the emphasis in setting the parameters has to be put on keeping a good conditioning. Since we

know the smallest distance between two nodes ($d_{\min}$), and that the nodes are near uniformly distributed, we can *normalize* the shape parameter as $\epsilon = \frac{\epsilon^*}{d_{\min}}$. In order to keep a good conditioning, we will increase parameter ϵ^* when the total number of nodes gets larger. A lot less work has been made in finding an optimal δ. In order to avoid level sets crossing, we set $\delta = d_{min} \times \delta^*$, where $0 < \delta^* < 1$. Figure 5 shows, for different values of δ^*, the l_∞ norm of the error, versus ϵ^*, in applying the Laplacian to the spherical harmonic $Y_{15}^3(x, y, z)$ on the unit sphere. We notice the error behavior described above, in which the error decreases with ϵ^* until the error abruptly grows due to the bad conditioning associated with very small shape parameters. We also notice that neither the value of δ nor the choice of the smooth radial function do seem to have a big influence on the error. For the remaining of this work, we will use the MQ RBF.

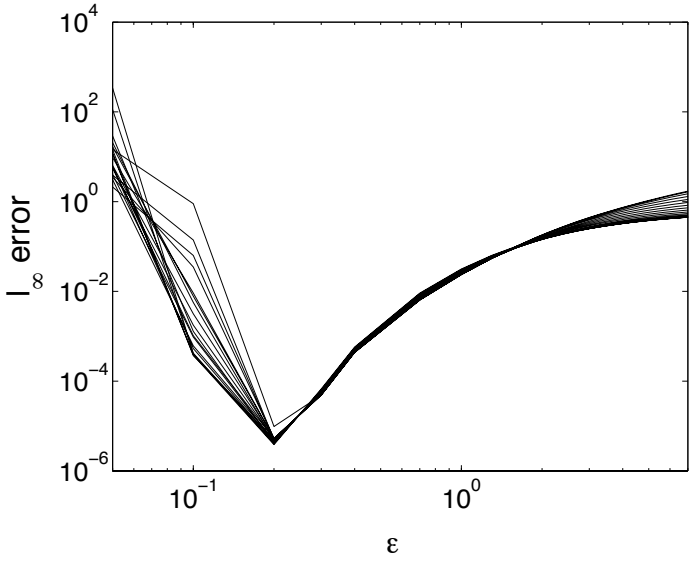

Fig. 5. Error in taking the surface Laplacian on the surface of the unit sphere of the spherical harmonic $Y_{15}^3(x, y, z)$, using 900 maximum determinant near uniformly distributed nodes. We show the error computed via the high order OGr method, for several values of the parameter δ, ranging from 0 to 1. We see that the effect of the parameter on the error is minimal.

5 Numerical Results

5.1 The Heat Equation

We solve the heat equation $u_t = \Delta_\Gamma u$ where Γ is the surface of the unit sphere. In Figure 6, we compare results from both the high order OGr and

the projection methods, with spherical harmonics of orders 1 and 3 as initial conditions. Since the eigenfunctions of the Laplace-Beltrami operator on the surface of the sphere are the spherical harmonics, we can easily analytically find the true solution to the problem and compute the error. It should be mentioned that the node distribution ranged from 64 to 900 nodes and that the solution was evaluated on a dense node set of 2500 points, on which the relative l_∞ error was finally computed.

The figure on the left also shows the error obtained on the same exercise via the Closest Point Method, reported in Table IV of [15]. We notice that the rate of error decay is about the same for both RBF methods. We note also that the decay seems to follow a straight line in the log-linear plots, indicating the expected spectral accuracy in both RBF methods. The error obtained via the Closest Point Method, however, decays algebraically as $O(h^2)$. This result makes sense since the OGr method is nothing but the Closest Point core idea with an RBF discretization instead of a FD discretization. We can therefore argue that, although the computational complexity of the Closest Point Method is smaller than the complexity of RBFs, a lot more nodes will be necessary than with RBFs, to obtain a same accuracy. For example, in Figure 6, we obtain an accuracy of about 5×10^{-5} with a node set of $h = 0.147$ with RBFs and with a node set of $h = 0.0125$ with the Closest Point Method. This means that, in order to obtain the same accuracy, one should use 64 nodes with RBFs and about $70,000$ nodes with the Closest Point Method. Furthermore, the time steps obtained via an RBF discretization are often much larger than time steps obtained via FD techniques. Thus, despite their very large computational complexity, RBFs would require all together a smaller amount of computations than the Closest Point Method. However, the scattered nature of the nodes makes it difficult to predict the exact rate of convergence of the RBF method, so the above calculation might not hold on different geometries or on different node sets, with different parameters. As for the superiority in accuracy of the Projection method over the OGr method in this particular instance (Figure 6), practice indicates that it should not be generalized to different geometries. For instance, the projection method only gives a lousy discretization of surface operators on rougher surfaces such as the femur in the following case.

The high order OGr method was used for the remainder of the test cases. Figure 7 shows the solution of the heat equation on the surface of a femur, at times $t = 0.04$, 0.08, 0.12 and 0.16. The initial condition is the gaussian bell $f(x, y, z) = e^{-4((x-x1)^2+(y-y1)^2+(z-z1)^2)}$, centered at $(x1, y1, z1)$.

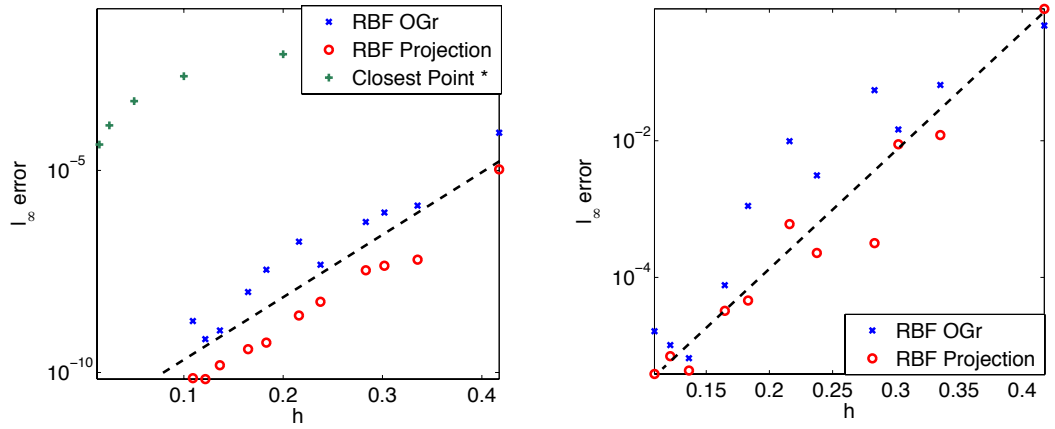

Fig. 6. Relative l_∞ error accrued after time t=1, using as an initial condition the spherical harmonic $Y_1^0(x, y, z)$ on the left and $Y_3^3(x, y, z)$ on the right. We show by the blue $\times$ the error produced when using the OGr method, and by the red o, the error produced by the projection method. We also show, by the green $+$, the error obtained on the same exercise via the Closest Point Method reported in Table IV of [15].

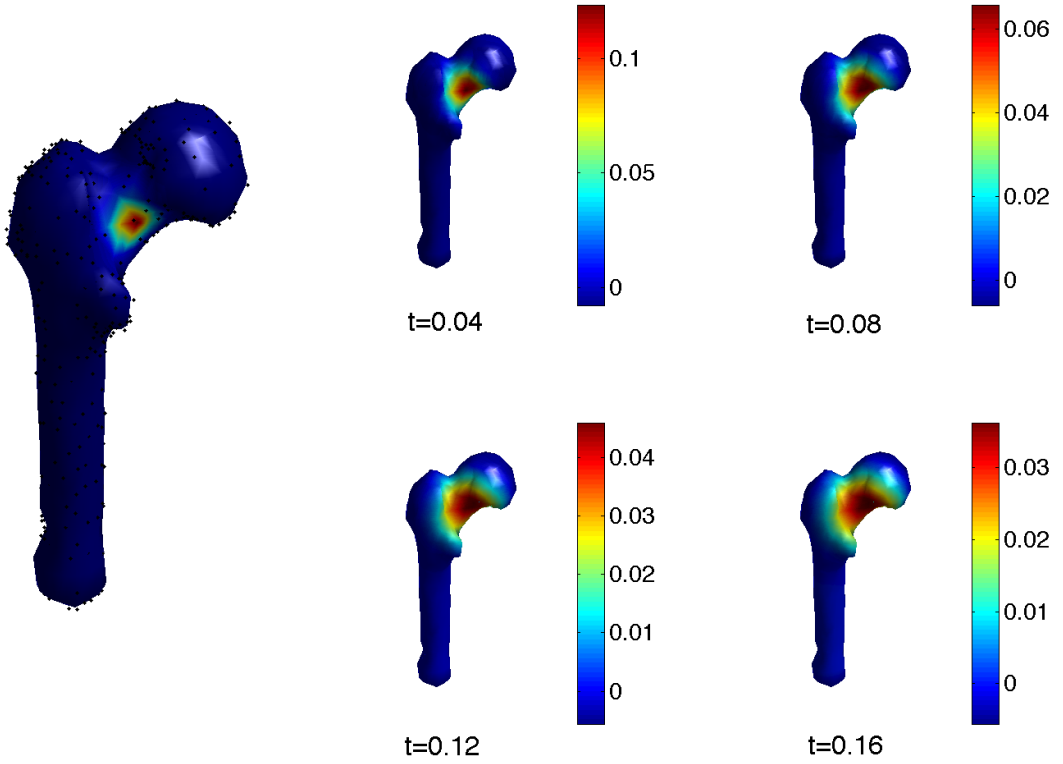

Fig. 7. Heat equation on the surface of a femur. The initial condition is the gaussian bell $f(x, y, z) = e^{-4((x-x1)^2+(y-y1)^2+(z-z1)^2)}$, centered at one of the nodes $(x1, y1, z1)$. The function at time $t = 0$ is shown on the left, along with the set of 512 nodes.

5.2 Eigenfunctions of the Laplace-Beltrami Operator

Finding a good discretization for the Laplace-Beltrami operator allows us to compute shape-dependent eigenfunctions and eigenvalues of the operator. In Figure 8, we show the torus and its distribution of 169 nodes, as well as the spectrum of the Laplace-Beltrami operator. We expect a set of purely real negative eigenvalues. We also reproduce Figure 4 from [44] in Figure 9, and show the eigenfunctions corresponding to λ_4 to λ_{15}.

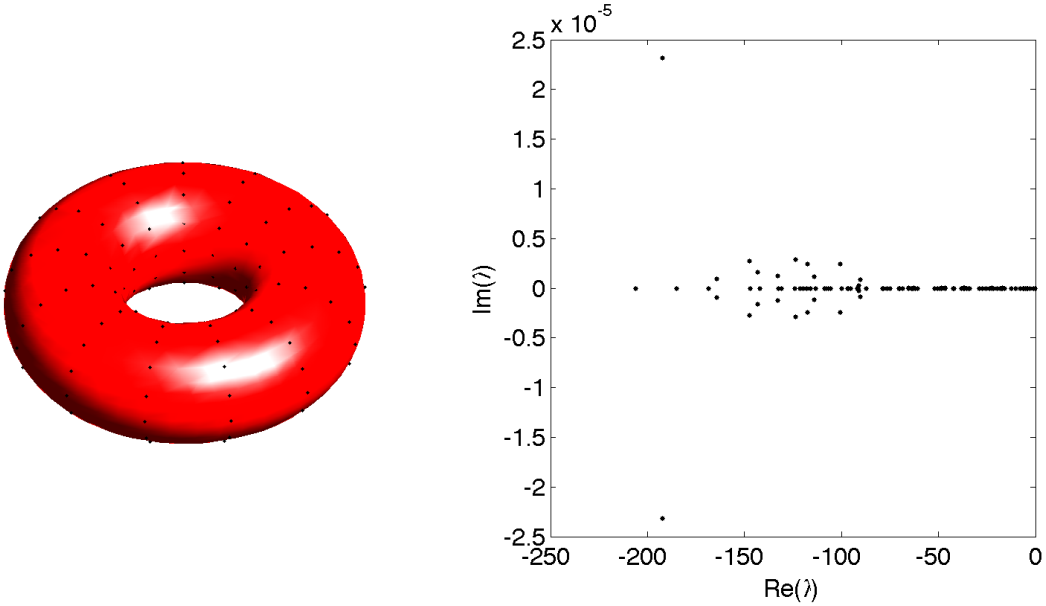

Fig. 8. The 169 nodes distributed on a torus are shown on the left, while the right picture displays the spectrum of the Laplace-Beltrami operator on the torus.

5.3 A Reaction-Diffusion Scheme

Reaction-diffusion theory has become an important field of research for its applications in morpheogenesis or epidemiology, ever since Alan Turing's 1952 paper [45] . The Brusselator is a non-linear reaction-diffusion scheme that was introduced in Brussels by Ilya Prigogine and his team. The Brusselator equations are the following

$$\begin{cases} \partial_t u = a - (b+1)u + u^2 v + D_u \Delta u \\ \partial_t v = bu - u^2 v + D_v \Delta v \end{cases}$$

In Figure 10, we display the steady state solution of the Brusselator on the surface of a frog, for two different sets of parameters.

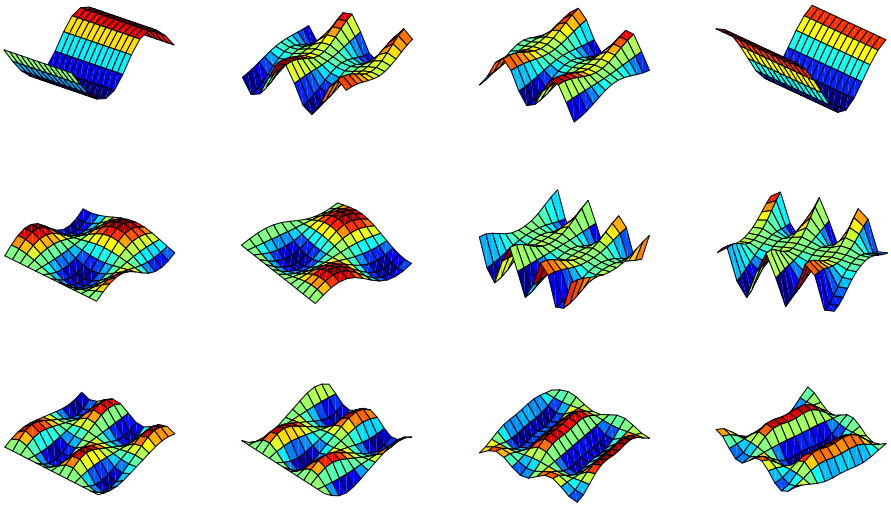

Fig. 9. We computed and display a reproduction of Figure 4 in [44], showing the eigenfunctions corresponding to λ_4 to $\lambda_1 5$.

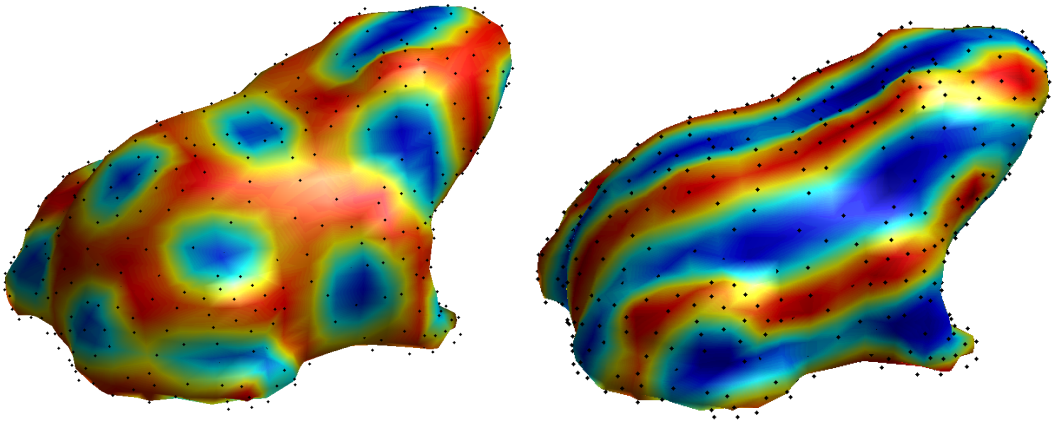

Fig. 10. Steady state solution of the Brusselator on the surface of a 560-node frog. The patterns are outcomes of two different sets of parameters in the Brusselator equations. (Figure on the left : du = 5, dv = 12, a = 4, b = 14. Figure on the right : du = 5, dv = 12, a = 3, b = 11)

5.4 An Application in Mesh Generation and Repair (using the software Gmsh)

Gmsh is an unstructured mesh generator [46]. It is sometimes difficult to create or repair mesh on complex manifolds. One way to produce a mesh is to create a discrete parametrization of the object, by solving Laplace's equation on its surface. In [47], the OGr method was shown to be a powerful technique and has been incorporated in Gmsh. Figures 11 and 12 show the mesh of a cat figurine respectively before and after the repair by our technique. In [47], we give the details of a mesh repair technique which consists in solving Laplace's equation on the surface with appropriate Dirichlet boundary condi-

tions, thereby finding a 1-to-1 map between the surface embedded in $\mathcal{R}^3$ and the unit disk. The mesh is then repaired on the unit disk and mapped back (via RBF interpolation) to the surface. It should be noted here that both the high version OGr and the Projection methods were coded and that the high version OGr gave significantly better results on most geometries.

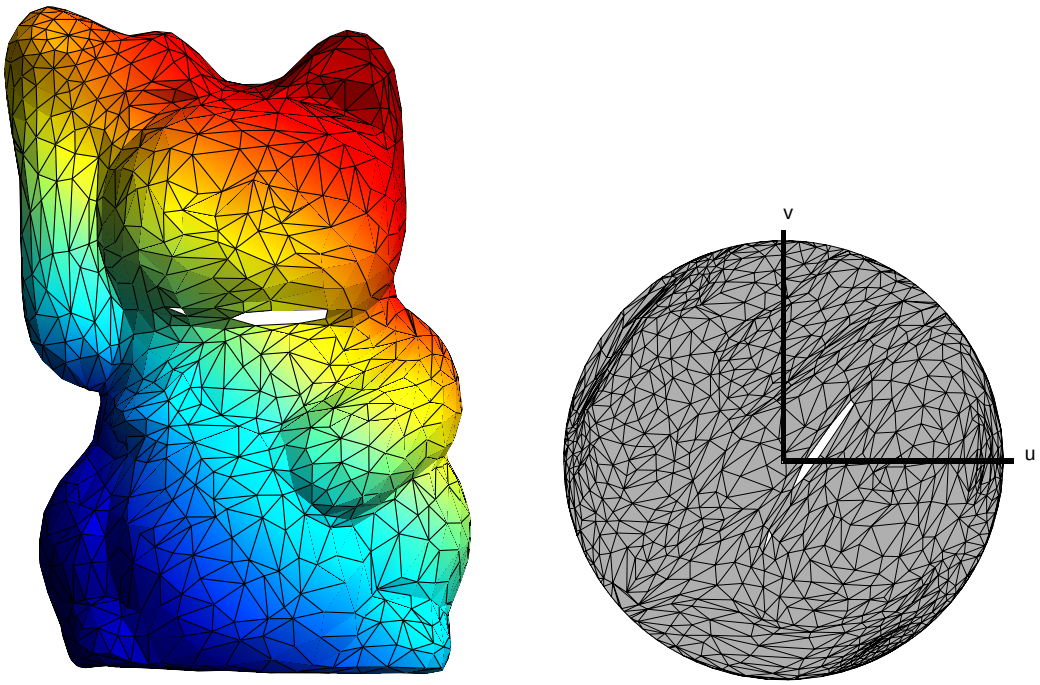

Fig. 11. A damaged mesh of a cat figurine is shown on the left. We want to find a 1-to-1 map between the surface embedded in $\mathcal{R}^3$ and the unit disk. We use the OGr method to find the solutions u and v of Laplace's equation with Dirichlet boundary conditions spanning both a period of the cosine function (displayed by the color on the left figure) and a period of the sine function. On the right, we plot the solutions (u, v) and therefore display the mesh mapped on the unit disk.

6 Conclusion

In this paper, we have introduced a new method, similar in essence to MacDonald and Ruuth's Closest Point Method, but brought to the realm of RBFs. We chose to define the surface implicitly via a level-set representation of three layers, as is often done in surface reconstruction problems. We are able, with only a point cloud given, to find an accurate discretization of the Laplace-Beltrami operator on the underlying surface. We showed the efficiency of the method on diverse problems, including on solving Laplace's equation on very complex geometries. This paper is intended to introduce the OGr method. Its numerical complexity limited us in the amount of nodes in each of the examples

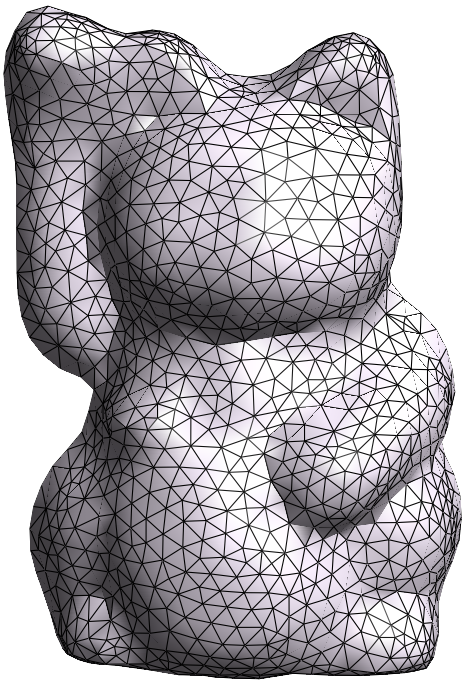

Fig. 12. Once the mesh has been mapped on the unit disk, it is repaired and mapped back to the original surface. This figure shows the repaired mesh of the cat figurine. More details on this procedure can be found in [47].

given. The surfaces described were therefore rather smooth. An adaptation of this method with fast algorithms, with different types of operators and in higher dimensions is in preparation. This upcoming version of the method will be able to handle much rougher features in the manifold, given that enough resolution is given in those areas.

Acknowledgements

The work of this author was supported by a FSR post-doctoral grant from the catholic University of Louvain. Part of the present work was conducted when the author was a Visiting Post-Doctoral Research Assistant at OCCAM (Oxford Centre for Collaborative Applied Mathematics) under support provided by Award No. KUK-C1-013-04 to the University of Oxford, UK, by King Abdullah University of Science and Technology (KAUST).

7 Appendix 1

We are showing here an alternate form of the differentiation matrix given in Section 4.1.2. We wish to impose the following

$$\begin{pmatrix} \Delta\Phi(\vec{x}) \\ \vec{n}\cdot\nabla\Phi(\vec{x})_{\vec{x}=\vec{x_i}} \\ (\vec{n}\cdot\nabla)(\vec{n}\cdot\nabla)\Phi(\vec{x})_{\vec{x}=\vec{x_i}} \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \nu \end{pmatrix} = \begin{pmatrix} \vec{g} \\ \vec{0} \\ \vec{0} \end{pmatrix}$$

The differentiation matrix takes the form

$$D = \begin{pmatrix} \Delta\Phi_x(x) \\ \vec{n}\cdot\nabla\Phi(\vec{x})_{\vec{x}=\vec{x_i}} \\ (\vec{n}\cdot\nabla)(\vec{n}\cdot\nabla)\Phi(\vec{x})_{\vec{x}=\vec{x_i}} \end{pmatrix} \begin{pmatrix} \Phi_x(x) & \Phi_{x^+}(x) & \Phi_{x^-}(x) \\ \Phi_x(x^+) & \Phi_{x^+}(x^+) & \Phi_{x^-}(x^+) \\ \Phi_x(x^-) & \Phi_{x^+}(x^-) & \Phi_{x^-}(x^-) \end{pmatrix}^{-1}$$

which can be use to solve Laplace's equation as follows

$$\begin{pmatrix} f_\Gamma \\ f_{\Gamma^+} \\ f_{\Gamma^-} \end{pmatrix} = \begin{pmatrix} D_{1,1} & D_{1,2} & D_{1,3} \\ D_{2,1} & D_{2,2} & D_{2,3} \\ D_{3,1} & D_{3,2} & D_{3,3} \end{pmatrix}^{-1} \begin{pmatrix} g_\Gamma \\ 0 \\ 0 \end{pmatrix}$$

Like before, one can reduce this equation to a system only containing values on the surface. One obtains $f_\Gamma = \tilde{D} g_\Gamma$.

8 Appendix 2

Below, we give the details on how to compute $\vec{n}\cdot\nabla\Phi(\vec{x})_{\vec{x}=\vec{x_i}}$ and $(\vec{n}\cdot\nabla)(\vec{n}\cdot\nabla)\Phi(\vec{x})_{\vec{x}=\vec{x_i}}$ from Section 4.1.2. Assume that we have an RBF expansion for the level-set distance function $s(\vec{x})$ whose zero-level set is the surface Γ. All the partial derivatives of radial functions can be computed analytically (see [34] for a few formulas). We can thus approximate and evaluate the derivatives

of s and of f quite easily by computing the derivatives of the radial functions.

$$\begin{cases} \vec{n} \cdot \nabla = s_x \partial_x + s_y \partial_y + s_z \partial_z \\ (\vec{n} \cdot \nabla)(\vec{n} \cdot \nabla) = (s_x)^2 \partial_{xx} + (s_y)^2 \partial_{yy} + (s_z)^2 \partial_{zz} \\ +2 s_x s_y \partial_{xy} + 2 s_x s_z \partial_{xz} + 2 s_y s_z \partial_{yz} \\ +(s_x s_{xx} + s_y s_{xy} + s_z s_{xz}) \partial_x \\ +(s_x s_{xy} + s_y s_{yy} + s_z s_{yz}) \partial_y \\ +(s_x s_{xz} + s_y s_{yz} + s_z s_{zz}) \partial_z \end{cases}$$

References

[1] N. Flyer, G. Wright, A radial basis function method for the shallow water equations on a sphere, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science 465 (2106) (2009) 1949–1976.

[2] E. Fuselier, G. Wright, Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates, SIAM J. Numer. Anal (2011), In Revision.

[3] A. Witkin, M. Kass, Reaction-diffusion textures, Computer Graphics 25 (1991) 299–308.

[4] J. Stam, Flows on surfaces of arbitrary topology, in: ACM Transactions On Graphics (TOG), Vol. 22, ACM, 2003, pp. 724–731.

[5] L. Lui, Y. Wang, T. Chan, Solving pdes on manifolds with global conformal parametrization, Variational, Geometric, and Level Set Methods in Computer Vision (2005) 307–319.

[6] M. Floater, K. Hormann, Surface parameterization: a tutorial and survey, Advances in multiresolution for geometric modelling (2005) 157–186.

[7] G. Turk, Generating textures on arbitrary surfaces using reaction-diffusion, in: ACM SIGGRAPH Computer Graphics, Vol. 25, ACM, 1991, pp. 289–298.

[8] M. Bertalmio, L. Cheng, S. Osher, G. Sapiro, Variational problems and partial differential equations on implicit surfaces, Journal of Computational Physics 174 (2001) 759–780.

[9] G. Dziuk, Finite elements for the beltrami operator on arbitrary surfaces, Partial Differential Equations and Calculus of Variations (1988) 142–155.

[10] G. Dziuk, C. Elliott, Surface finite elements for parabolic equations, Journal of Computational Mathematics 25 (4) (2007) 385–407.

[11] Q. Du, L. Ju, L. Tian, Finite element approximation of the cahn-hilliard equation on surfaces, Computer Methods in Applied Mechanics and Engineering.

[12] F. Mémoli, G. Sapiro, P. Thompson, Implicit brain imaging, Neuroimage 23 (Supplement 1) (2004) 179–188.

[13] J. Greer, An improvement of a recent eulerian method for solving pdes on general geometries, Journal of Scientific Computing 29 (3) (2006) 321–352.

[14] J. Greer, Fourth order partial differential equations on general geometries, Tech. rep., DTIC Document (2005).

[15] S. Ruuth, B. Merriman, A simple embedding method for solving partial differential equations on surfaces, Journal of Computational Physics 227 (3) (2008) 1943–1961.

[16] B. Merriman, S. Ruuth, Diffusion generated motion of curves on surfaces, Journal of Computational Physics 225 (2) (2007) 2267–2282.

[17] L. Tian, C. Macdonald, S. Ruuth, Segmentation on surfaces with the closest point method, in: Image Processing (ICIP), 2009 16th IEEE International Conference on, IEEE, 2009, pp. 3009–3012.

[18] C. Macdonald, S. Ruuth, Level set equations on surfaces via the closest point method, Journal of Scientific Computing 35 (2) (2008) 219–240.

[19] C. Macdonald, J. Brandman, S. Ruuth, Solving eigenvalue problems on curved surfaces using the closest point method, Journal of Computational Physics 230 (22) (2011) 7944–7956.

[20] R. Hardy, Multiquadric equations of topography and other irregular surfaces, Journal of Geophysical Research 76 (8) (1971) 1905–1915.

[21] R. Franke, Scattered data interpolation: Tests of some methods., Math. Comput. 38 (157) (1982) 181–200.

[22] C. Micchelli, Interpolation of scattered data: distance matrices and conditionally positive definite functions, Constructive Approximation 2 (1) (1986) 11–22.

[23] A. Witkin, P. Heckbert, Using particles to sample and control implicit surfaces, Computer Graphics 28 (Annual Conference Series) (1994) 269–277.

[24] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, Computer graphics -New York-Association for computing machinery 26 (1992) 71–71.

[25] V. Savchenko, A. Pasko, O. Okunev, T. Kunii, Function representation of solids reconstructed from scattered surface points and contours, in: Computer Graphics Forum, Vol. 14, Wiley Online Library, 1995, pp. 181–188.

[26] G. Turk, J. O'Brien, Variational implicit surfaces, Techical Reports GIT-GVU-99-15.

[27] G. Turk, J. O'Brien, Shape transformation using variational implicit functions, Computer Graphics 33 (Annual Conference Series) (1999) 335–342.

[28] G. Turk, J. O'brien, Modelling with implicit surfaces that interpolate, ACM Trans. Graph. 21 (4) (2002) 855–873.

[29] B. Morse, T. Yoo, P. Rheingans, D. Chen, K. Subramanian, Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions, in: Shape Modeling and Applications, SMI 2001 International Conference on., IEEE, 2001, pp. 89–98.

[30] Y. Ohtake, A. Belyaev, H. Seidel, A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions, in: Proceedings of the Shape Modeling International 2003, IEEE Computer Society, 2003, pp. 153–161.

[31] A. Corrigan, H. Dinh, Computing and rendering implicit surfaces composed of radial basis functions on the gpu, in: Poster proceedings of the International Workshop on Volume Graphics, 2005.

[32] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum, T. Evans, Reconstruction and representation of 3d objects with radial basis functions, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, 2001, pp. 67–76.

[33] J. Carr, R. Beatson, B. McCallum, W. Fright, T. McLennan, T. Mitchell, Smooth surface reconstruction from noisy range data, ACM GRAPHITE 3 (2003) 119–126.

[34] G. Fasshauer, Meshfree approximation methods with MATLAB, Interdisciplinary mathematical sciences, World Scientific, 2007.

[35] H. Wendland, Fast evaluation of radial basis functions: Methods based on partition of unity, in: Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt University Press, 2002, pp. 473–483.

[36] E. Kansa, Multiquadrics–a scattered data approximation scheme with applications to computational fluid-dynamics–ii solutions to parabolic, hyperbolic and elliptic partial differential equations, Computers & mathematics with applications 19 (8-9) (1990) 147–161.

[37] E. Kansa, Multiquadrics–a scattered data approximation scheme with applications to computational fluid-dynamics–i surface approximations and partial derivative estimates, Computers & Mathematics with Applications 19 (8-9) (1990) 127–145.

[38] B. Fornberg, C. Piret, On choosing a radial basis function and a shape parameter when solving a convective pde on a sphere, Journal of Computational Physics 227 (5) (2008) 2758–2780.

[39] C. Elliott, B. Stinner, V. Styles, R. Welford, Numerical computation of advection and diffusion on evolving diffuse interfaces, IMA Journal of Numerical Analysis 31 (3) (2011) 786–812.

[40] B. Fornberg, G. Wright, E. Larsson, Some observations regarding interpolants in the limit of flat radial basis functions, Computers & Mathematics with Applications 47 (1) (2004) 37–55.

[41] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl. 48 (2004) 853–867.

[42] B. Fornberg, C. Piret, A stable algorithm for flat radial basis functions on a sphere, SIAM J. Sci. Comput 30 (1) (2007) 60–80.

[43] B. Fornberg, E. Larsson, N. Flyer, Stable computations with gaussian radial basis functions, SIAM Journal of Scientific Computing 33 (2011) 869–892.

[44] R. Glowinski, D. Sorensen, Computing the eigenvalues of the laplace-beltrami operator on the surface of a torus: A numerical approach, Partial Differential Equations (2008) 225–232.

[45] A. Turing, The chemical basis of morphogenesis, Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 237 (641) (1952) 37–72.

[46] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.

[47] E. Marchandise, C. Piret, J.-F. Remacle, Cad and mesh repair with radial basis functions., J. Comput. Physics 231 (5) (2012) 2376–2387.