Simple computation of reaction–diffusion processes on point clouds

Colin B. Macdonald[a], Barry Merriman[b], and Steven J. Ruuth[c,1]

[a]Mathematical Institute, University of Oxford, Oxford OX1 3LB, United Kingdom; [b]Department of Human Genetics, University of California, Los Angeles, CA 90095; and [c]Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada V5A 1S6

Edited* by Stanley Osher, University of California, Los Angeles, CA, and approved April 15, 2013 (received for review December 7, 2012)

The study of reaction–diffusion processes is much more complicated on general curved surfaces than on standard Cartesian coordinate spaces. Here we show how to formulate and solve systems of reaction–diffusion equations on surfaces in an extremely simple way, using only the standard Cartesian form of differential operators, and a discrete unorganized point set to represent the surface. Our method decouples surface geometry from the underlying differential operators. As a consequence, it becomes possible to formulate and solve rather general reaction–diffusion equations on general surfaces without having to consider the complexities of differential geometry or sophisticated numerical analysis. To illustrate the generality of the method, computations for surface diffusion, pattern formation, excitable media, and bulk-surface coupling are provided for a variety of complex point cloud surfaces.

closest point method | embedding method | Laplace–Beltrami

Partial differential equations (PDEs) are widely used to describe continuum processes such as diffusion, chemical reactions, fluid flow, or electrodynamics. In standard 3D settings, these take a familiar PDE form, such as a reaction–diffusion equation:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + f(u),$$

and the ways to numerically solve such equations are well-developed. The basic approach can be quite simple, such as laying down a uniform Cartesian grid of points, $\{\boldsymbol{x}_{i,j,k} : 1 \le i,j,k \le N\}$, and using simple, familiar approximations of the differential terms on this grid, such as:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{h^2},$$

where $u_{i,j,k} \approx u(\boldsymbol{x}_{i,j,k})$ and h is the uniform grid spacing. As a result, it is very easy to implement methods for the numerical solution of such equations to study the phenomena of interest. To achieve efficiency for large-scale computations, more advanced methods are required, such as implicit discretization and solvers for systems of equations. These solvers, while internally complex, have been implemented in standard, well-validated numerical routines and are accessible through numerical subroutine libraries.

Important physical processes also arise in complex geometrical settings, such as on complicated surfaces in three dimensions. The abstract form of differential operators on surfaces remains the same as in 3D, however, when explicitly expressed in coordinates, the formulas for the operators and the corresponding discretized expressions are relatively complicated and have received much less attention. Moreover, in practical settings, surfaces are often defined simply as a set of points—a point cloud—sampled from the underlying surface. Because the connectivity of the points is not provided, this adds further complexity to methods that need to reconstruct the geometric properties of the surface, such as the metric distance.

Here we present a method for solving reaction–diffusion equations on a point cloud that represents the underlying surface—or any other geometric object—in a way that reduces the problem to working with entirely standard classical 3D discretizations and solver libraries. Our approach is fundamentally different from other recent methods for computing point clouds (e.g., refs. 1, 2), in that it does not make assumptions on the codimension of the set upon which we compute, nor does it ever construct the local connectivity between points (cf. ref. 2). A more detailed comparison with other methods is provided in the *Discussion* section.

The article unfolds as follows: We begin by introducing the problem and reviewing some recent results that are relevant for our method. An algorithm for solving diffusion on point cloud surfaces is developed by formulating the surface PDE problem in the embedding space and by giving a convolutional discretization of the corresponding equations. Numerical experiments are provided to explore the accuracy and stability of the method. An extension to reaction–diffusion equations is developed, and some experiments on systems are provided. The article concludes by contrasting the method to some of the others available, and outlining generalizations and directions for future work.

Diffusion Intrinsic to a Surface

We develop the method first for the fundamental case of pure diffusion.

Consider evolving a scalar diffusion process on a smooth surface $\mathcal{S}$. The corresponding PDE is, formally:

$$u_t = \nabla_{\mathcal{S}}^2 u, \qquad [1]$$

where the right-hand side involves the Laplacian intrinsic to the surface, also known as the Laplace–Beltrami operator. This generalizes the standard Laplacian, an operator with a well-known Cartesian coordinate formula, to a surface. We assume that in practice we do not know $\mathcal{S}$; instead, we are given a discrete, unconnected set of points $\mathcal{C}$ which sample $\mathcal{S}$ (perhaps even including noise; i.e., error in point positions relative to the true geometry). We seek a method that solves Eq. **1** without explicitly constructing an approximation to $\mathcal{S}$ from the point cloud $\mathcal{C}$, and which uses only knowledge of how to express and solve the standard 3D diffusion equation in standard 3D Cartesian space.

Continuous Formulation in the Embedding Space. Our first step to formulating a method is to replace the surface PDE by a related equation posed on the surrounding 3D space that can be solved using standard Cartesian grid methods in 3D. When restricted to the surface, this 3D embedding equation will give the solution to the original surface problem. To obtain an intuition for the design of this equation, consider a function u defined on a smooth

Author contributions: C.B.M., B.M., and S.J.R. designed research; C.B.M. and S.J.R. performed research; C.B.M. and S.J.R. contributed new reagents/analytic tools; C.B.M., B.M., and S.J.R. analyzed data; and C.B.M., B.M., and S.J.R. wrote the paper.

The authors declare no conflict of interest.

*This Direct Submission article had a prearranged editor.

[1]To whom correspondence should be addressed. E-mail: sruuth@sfu.ca.

surface. The function u may be extended to a 3D neighborhood of the surface by extending its value constant along the normal directions to the surface. The 3D gradients of this extended function agree with the surface gradients of the surface function at the surface, since the only 3D variation is along the surface. Expressing this intuition mathematically leads to the following fundamental principle (3):

1. Equivalence of gradients. Suppose u is any function defined on $\mathbb{R}^3$ that is constant along directions normal to the surface. Then, at the surface:

$$\nabla u = \nabla_{\mathcal{S}} u,$$

where $\nabla_{\mathcal{S}}$ denotes the intrinsic surface gradient [specifically, $\nabla_{\mathcal{S}} u$ is the tangent vector to the surface defined by $\nabla_{\mathcal{S}} u \equiv \nabla u - \hat{n}(\hat{n} \cdot \nabla u)$, where $\hat{n}$ is a unit normal to the surface].

Similarly, a flux that is everywhere directed along the surface can only spread out within the surface directions. This gives us our second fundamental principle (3):

2. Equivalence of divergence. Suppose v is any vector field on $\mathbb{R}^3$ that is tangent at $\mathcal{S}$, and also tangent at all surfaces displaced by a fixed distance from $\mathcal{S}$ (i.e., all surfaces defined as level sets of the distance function to $\mathcal{S}$). Then, at the surface:

$$\nabla \cdot v = \nabla_{\mathcal{S}} \cdot v.$$

In fact, a version of this principle is true even if the vector field v is not tangent to $\mathcal{S}$ (4).

In both instances, we observe that derivatives intrinsic to the surface may be computed in 3D by extending quantities off the surface along the normal directions emanating from the surface. Our approach for numerically approximating PDEs on surfaces, the closest point method (3), evaluates derivatives using these fundamental principles and a "closest point function" representation of the surface $\mathcal{S}$: for a point $\boldsymbol{x}$ in space, let $\mathrm{cp}_{\mathcal{S}}(\boldsymbol{x})$ be the point belonging to $\mathcal{S}$ that is closest to $\boldsymbol{x}$. Notice that the constant normal extension may be efficiently evaluated by interpolation at the closest point on the surface, and that the closest point representation makes sense for any point set $\mathcal{S}$, not just a continuous surface. The latter property is what will give our approach its geometric generality to point clouds.

Combining these results gives the following theorem for evaluating the Laplace–Beltrami operator using only the classical 3D Cartesian Laplacian and the closest point function (3):

Theorem 1. *Let $\mathcal{S}$ be a smooth surface in $\mathbb{R}^3$ and $u : \mathcal{S} \to \mathbb{R}$ be a smooth function. Let $\mathrm{cp}_{\mathcal{S}}(\boldsymbol{x})$ be the closest point function for $\mathcal{S}$. Then:*

$$\nabla^2_{\mathcal{S}} u(\boldsymbol{x}) = \nabla^2 u(\mathrm{cp}_{\mathcal{S}}(\boldsymbol{x})) \quad \text{for } \boldsymbol{x} \in \mathcal{S}. \qquad [2]$$

In addition, this remains true in the general case where $\mathcal{S}$ is a smooth object of dimensions $k \le d$ in $\mathbb{R}^d$, such as for a filament ($k = 1$) in $\mathbb{R}^3$ (3, 4).

The right-hand side of Eq. **2** is well-defined for a surface $\mathcal{S}$ because $u(\mathrm{cp}_{\mathcal{S}}(\cdot))$ is a function on $\mathbb{R}^3$ over which the standard Laplacian is defined. A consequence of this is that Eq. **2** will provide a generalization of the Laplace operator to point clouds, as well as a practical means of computing the corresponding diffusion flows.

Applying Eq. **2** to the surface PDE (Eq. **1**) yields the explicit closest point method for solving the surface PDE in the embedding space. In its simplest form, the method alternates two steps. First, quantities on the surface are extended into the embedding space using the closest point operator:

$$v = u(\mathrm{cp}_{\mathcal{S}}). \qquad [3]$$

Next, a short-time evolution of the corresponding PDE in 3D is carried out:

$$u_t = \nabla^2 u, \qquad [4]$$

starting from the extended variable, v. For a small time, the solution of this embedding equation gives the desired approximation of the underlying surface flow (Eq. **1**) on the surface $\mathcal{S}$.

To illustrate the intuition behind the method, consider a surface process that is driven by diffusion. Suppose the evolving function is artificially extended to all of $\mathbb{R}^3$ by making it constant along the normal directions off the surface. We then apply standard 3D diffusion to the extended function for a short time dt. Intuitively, the fully 3D diffusion is driven by in-surface gradients since the extended function does not vary normal to the surface. This implies that the method will carry out the desired in-surface flow, at least for the short time scale of interest. Note that the leading order term in the error is $O(dt)$ since the second derivative of u normal to the surface is only guaranteed to be zero initially; at later times, it will vary linearly with dt.

As noted, the closest point method is valid for smooth surfaces of any codimension embedded in $\mathbb{R}^d$. In particular, in $\mathbb{R}^3$, it applies to 2D surface diffusion as well as 1D diffusion along a filament. It is even valid for solid regions since Eqs. **3** and **4** reduce to the standard heat equation in $\mathbb{R}^d$ when applied to a d-dimensional subset of $\mathbb{R}^d$. Because the extension procedure is well-defined and easily performed even if the surface is simply represented as a point cloud, and because the evolution takes place in standard 3D, the method retains the same simplicity no matter how complex the surface or point set on which the problem is posed. The discretization of the method that we develop preserves this geometric flexibility, and ultimately yields methods that are independent of the dimensionality of the underlying point cloud set.

Discretization of the Embedding Equation. The closest point method must be discretized in time and space. To construct the discretization, it is crucial to note that we do not have $\mathcal{S}$; instead, we have a point cloud approximation $\mathcal{C}$ of the surface. Consequently, we may assume that the easily computed $\mathrm{cp}_{\mathcal{C}}(\boldsymbol{x})$ is available, but not $\mathrm{cp}_{\mathcal{S}}(\boldsymbol{x})$. Specifically, let $\mathrm{cp}_{\mathcal{C}}(\boldsymbol{x})$ be the point belonging to the set $\mathcal{C}$ that is closest to $\boldsymbol{x}$.

To carry out the first step in the closest point method, we make a direct replacement of $\mathrm{cp}_{\mathcal{S}}(\boldsymbol{x})$ by $\mathrm{cp}_{\mathcal{C}}(\boldsymbol{x})$ in Eq. **3**. That is, the constant normal extension is carried out by replacing u in the embedding space by the value at the closest point in the point cloud, $u(\mathrm{cp}_{\mathcal{C}}(\boldsymbol{x}))$. Similar to refs. 3, 5, we carry out this step on a uniform grid with spacing h in a dimension-by-dimension fashion using standard polynomial interpolation of degree $p \ge 3$.

In refs. 3, 5 standard finite differences are used in the second step of the closest point method to discretize the 3D diffusion flow (Eq. **4**). Unfortunately, this gives poor accuracy and is not recommended except for very dense point clouds. To see this, observe that $u(\mathrm{cp}_{\mathcal{C}}(\boldsymbol{x})) = u(\mathrm{cp}_{\mathcal{S}}(\boldsymbol{x})) + O(\delta)$ on the surface for a smooth $u(\cdot)$ where δ is the maximum distance between an arbitrary point on the surface and its closest point in the point cloud. Forming the discrete Laplacian amplifies the error by a factor of $1/h^2$, where h is the mesh spacing, potentially leading to an $O(\delta/h^2)$ error at each time step.

To avoid the amplification of errors arising in a standard finite difference approach, we discretize the 3D diffusion flow (Eq. **4**) convolutionally; that is, we convolve the extended function with a kernel corresponding to diffusion for a short time dt. The convolution is evaluated in Fourier space via the fast Fourier transform (FFT). We remark that to apply our discretization the problem must have a convolutional form and the corresponding convolution kernel should be accurately approximated using the discrete Fourier basis. This article focuses on flows involving diffusion and we note that there will be situations where our discretizations do not apply (e.g., pure convection).

In a typical point cloud calculation, we are given a cloud of points and choose a suitable time step-size dt. Normally, the time step-size should be chosen so that the effective width of the

diffusion kernel (namely $2\sqrt{dt}$) exceeds the distance between points in the cloud. If this condition is violated, diffusion is insufficiently evolved to cause the desired coupling between neighboring points and the method assumes that there is a hole in the surface. We may quantify this condition by assigning a measure μ to the point cloud spacing:

$$\mu = \max_i \min_{j \neq i} \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2, \quad [5]$$

where the points $\boldsymbol{p}_i, 1 \leq i \leq M$ define a cloud of M points. With this measure of spacing, we see that we want to choose dt such that $2\sqrt{dt} > \mu$. The grid spacing h is determined from dt. Specifically, we select a mesh that is sufficiently refined to ensure that the error arising from truncating the corresponding Fourier series is exponentially small. This leads us to the requirement $\frac{h^2}{4dt} \ll 1$, which is a standard choice in spectral convolution (6).

Estimates of the errors arising from one step of the method may now be formally computed. We have already seen that alternating diffusion for a time dt with constant normal extension leads to an $O(dt^2)$ error after one time step of size dt. The replacement of $\mathrm{cp}_S(\boldsymbol{x})$ by $\mathrm{cp}_C(\boldsymbol{x})$ in Eq. **3** introduces an $O(\delta)$ error into the approximation of v. Interpolation with degree-p polynomials introduces an additional $O(h^{p+1})$ error into the approximation of v. Using $p \geq 3$ and $\frac{h^2}{4dt} \ll 1$, we find that this interpolation error is $O(dt^2)$. Finally, errors arise from truncating the Fourier series. These are exponentially small relative to the other errors induced by the method provided $\frac{h^2}{4dt} \ll 1$.

Methods. The closest point function is constructed using the k-nearest neighbors algorithm. An implementation of this step in the high-level language MATLAB is given by:

```
ind = knnsearch(PT_CLOUD, [X(:) Y(:) Z(:)]);
CPx = reshape(PT_CLOUD(ind, 1), size(X));
CPy = reshape(PT_CLOUD(ind, 2), size(Y));
CPz = reshape(PT_CLOUD(ind, 3), size(Z));
```

where PT_CLOUD is an $M \times 3$ matrix defining a cloud of M points in 3D; X, Y, Z are matrices specifying the x, y, z coordinates of the grid nodes; and CPx, CPy, CPz are the corresponding coordinates of the closest points in the point cloud.

To evolve diffusion on point cloud surfaces, we alternate two steps. The first step extends values off the surface into a 3D region surrounding the surface using tricubic interpolation. This step can be implemented in MATLAB using:

```
V = interp3(X, Y, Z, U, CPx, CPy, CPz, 'cubic');
```

The second step simply evolves the corresponding 3D PDE in a neighborhood of the surface. In MATLAB, this can be accomplished on a uniform lattice using:

```
U = ifftn(K_hat .* fftn(V));
```

where convolution is carried out using K_hat, the discrete Fourier transform of the diffusion kernel. Note that fftn and ifftn are the standard 3D Fourier transform and inverse Fourier transform in MATLAB. Thus, the only difference between this code and a standard one for solving diffusion equations in 3D is the constant normal extension of the quantity U preceding each time step. In our code we compute the K_hat matrix analytically; for example, on a cube of side length L with a grid of Nx = L/h points in each direction, diffusion coefficient D, and step size dt, we have:

```
k = fftshift((2 * pi/L) * [-Nx/2 : Nx/2-1]);
[kx, ky, kz] = meshgrid(k, k, k);
K_hat = exp(-(kx.^2 + ky.^2 + kz.^2) * D * dt);
```

In this work, we use a uniform grid of the embedding space $\mathbb{R}^3$. This approach is simple and makes direct use of the built-in MATLAB functions for cubic interpolation and FFTs. For improved efficiency, interpolation can be carried out in a band around the surface since the diffusion kernel becomes exponentially small outside of its effective width (see refs. 3, 5 for localization techniques for the finite difference case). Furthermore, the convolution step may be accelerated using the fft() and ifft() functions on a graphics processing unit. Even with these enhancements, some computations will remain computationally expensive. For example, it is expensive to apply this discretization to a PDE on a surface that is embedded in four or more dimensions. It may also be inefficient to compute on a uniform discretization of space if the point cloud sampling of the underlying surface is highly nonuniform. For such problems, alternative methods for evaluating the convolutions are needed. A particularly promising approach is to evaluate convolutions by a suitable spatial quadrature formula, and compute the subsequent discrete sums by the fast Gauss transform (7, 8). This fast convolution technique requires only $O(N)$ operations per time step, where N is the number of grid nodes forming the discretization.

Numerical Experiment. We consider an initial value problem for diffusion on the unit sphere with exact solution $u(\theta, \eta, t) = \exp(-2t)\cos(\eta)$ in spherical coordinates (r, θ, η).

The evolution is carried out on a point cloud defined by the method of Saff and Kuijlaars (9). This method places nodes along a spiral in such a way that the distance between nodes along the spiral is approximately equal to the distance between the coils of the spiral. Using cubic interpolation, the max norm relative errors at $T_f = 0.1$ are computed for a variety of point clouds. This yields the results reported in Table 1. Our experiments measure the point cloud spacing using the value μ given by Eq. **5**. This convergence test indicates a first order error in μ, and shows that the regularity of the error improves as the discretization parameters dt and h are refined.

Reaction–Diffusion Intrinsic to a Surface

Many surface processes are modeled by systems of reaction–diffusion equations. For illustrative purposes, this article considers systems of two equations:

$$u_t = f(u, v) + D_u \nabla_S^2 u, \; v_t = g(u, v) + D_v \nabla_S^2 v.$$

Applying a forward Euler discretization for the reaction terms and a backward Euler discretization for the diffusion terms yields:

$$u^{n+1} = u^n + dt \cdot f(u^n, v^n) + dt \cdot D_u \nabla_S^2 u^{n+1}$$
$$v^{n+1} = v^n + dt \cdot g(u^n, v^n) + dt \cdot D_v \nabla_S^2 v^{n+1}.$$

This decouples the equations for u^{n+1} and v^{n+1}. Similar to the scalar diffusion case, we alternate between a convolutional

Table 1. Max norm relative errors for the heat equation on a sphere.

μ	$h = 1/16, dt = T_f/8$	$h = 1/64, dt = T_f/16$
0.11025	0.0622	0.0728
0.05585	0.0346	0.0326
0.02792	0.0154	0.0152
0.01401	0.0087	0.0075
—	—	—
0	0.0021	0.0010

discretization and a reextension using tricubic interpolation. In MATLAB, the system can be evolved to time $N \cdot dt$ using:

```
for i = 1:N
  U_ = ifftn(Ku_hat .* fftn(U + dt * f(U, V)));
  V_ = ifftn(Kv_hat .* fftn(V + dt * g(U, V)));
  U = interp3(X, Y, Z, U_, CPx, CPy, CPz, 'cubic');
  V = interp3(X, Y, Z, V_, CPx, CPy, CPz, 'cubic');
end
```

where Ku_hat and Kv_hat are the discrete Fourier transforms of the diffusion kernels for $u_t = D_u \nabla_S^2 u$ and $v_t = D_v \nabla_S^2 v$, respectively. As in the scalar diffusion case, X, Y, Z are matrices specifying the x, y, z coordinates of the grid nodes, and CPx, CPy, CPz are the corresponding coordinates of the closest points in the point cloud.

Patterns on Surfaces. Fig. 1 displays the result of a computation for the Gray–Scott model (10) for pattern formation (11). This model has reaction terms: $f(u,v) = -uv^2 + F(1-u)$, $g(u,v) = uv^2 - (F+k)v$. The parameters are chosen to be $F = 0.054$, $k = 0.063$, $D_u = 1/8100$, and $D_v = D_u/2$. The computation starts with a perturbation of the steady-state values of u and v localized near the right of the *Drosophila* embryo, and continues until $t = 7,000$. By choosing this as our final time, we stop the evolution before the entire surface is covered in stripes. The *Drosophila* embryo surface (12) consists of 5,709 cells, which form a point cloud about four units long. We discretize using a $120 \times 48 \times 48$ lattice on a $5 \times 2 \times 2$ box and select a time step-size of $dt = 10$. In our visualization, each point in the point cloud is represented by a small sphere colored according to the value of the solution u.

Fig. 2 also shows pattern formation, in this case for the Brusselator (13), a model that has reaction terms: $f(u,v) = a - (b+1)u + u^2 v$, $g(u,v) = bu - u^2 v$. To give interesting steady patterns, the parameters are chosen to be $a = 3$, $b = 10.2$, $D_u = 21/6,480$, and $D_v = 50/6,480$. The initial conditions are perturbations around the steady state. The sea shell is a surface represented by a cloud of 39,733 points (14), and is about 2.2 units wide. A lattice of $32 \times 48 \times 24$ points and a time step-size of 0.1 is used. In this example and the excitable media example below, visualizations are obtained using MATLAB's trisurf() command and a known triangulation. We emphasize, however, that a triangulation is not used in the PDE solver itself. Note that effective visualization methods are available when surface triangulations are unavailable (see ref. 15 for some examples). It is also possible to perform real-time ray-casting on the closest point representation of the surface cp_S (16), although we have not explored this option for point clouds using cp_C.

Patterns on Solid Objects. Fig. 3 displays another result for the Brusselator model with these same choices of parameters and initial conditions. However, the point cloud in this example consists of points selected randomly between spheres of radii 0.5 and 1. The same code is used for this 3D solid region computation as for our 2D surface computations. Zero flux boundary conditions naturally arise, an effect that leads to stripes intersecting the boundaries at about 90°. Dirichlet boundary conditions may alternatively be imposed; for such problems, the known boundary values are propagated out as part of the extension step. The analytical form of the shape is not used to compute the solution; it is only used as part of the visualization. A $200 \times 200 \times 200$ lattice is used with a time step-size of 0.1.

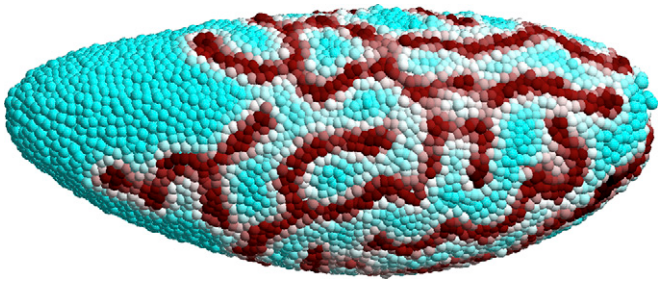

Fig. 1. A Turing pattern on a point cloud *Drosophila* embryo surface (5,709 points).

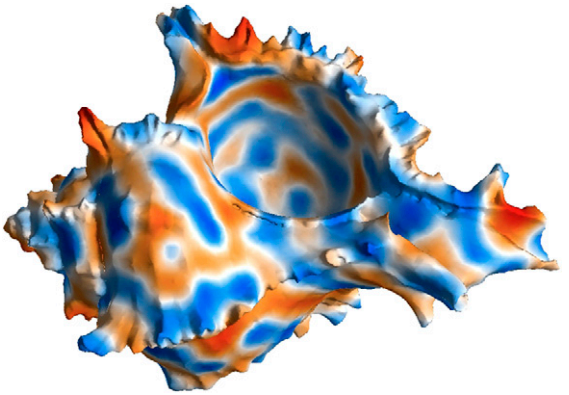

Fig. 2. A Turing pattern on a point cloud sea shell (39,733 points).

This example considers a reaction–diffusion equation with homogeneous Neumann boundary conditions on a densely sampled 3D spherical shell. Suppose we consider the corresponding family of problems on solid spherical shells that vary by thickness. In the limit as the thickness tends to zero, we obtain a reaction–diffusion equation intrinsic to a sphere. It is interesting to note that our approach can compute the solution to this entire family of problems, including the limiting case on a sphere, with the same code and without imposing any assumptions on the dimensionality of the object upon which we compute.

Excitable Media. Fig. 4 is the result of a computation for the Fitzhugh–Nagumo model (17, 18), which is a model for excitable media (19). This model has reaction terms: $f(u,v) = (a-u)(u-1)u - v$, $g(u,v) = \epsilon(\beta u - \gamma v - \delta)$. To obtain traveling waves, appropriate parameters and initial conditions must be used. We take $a = 0.1, \epsilon = 0.015, \beta = 0.5$, $\gamma = 1.0, \delta = 0, D_u = 1.44 \times 10^{-4}$ and $D_v = 3.6 \times 10^{-6}$. For initial conditions, the variable u is set equal to 1 for $x, y, z > 0$ and 0 elsewhere, while v is set equal to 1 for $x < 0, y > 0, z > 0$ and 0 elsewhere. The surface, the iconic heart (20), is represented by a point cloud of 57,346 points. It is centered at the origin and is about 2.15 units in height. A lattice of $160 \times 160 \times 160$ points with a time step-size of 0.875 is used.

Bulk Coupling. Our method also solves surface reaction–diffusion equations that are coupled to processes occurring in the bulk surrounding space. Typically, the surface is coupled to the bulk through the boundary conditions of the bulk equation.

Suppose we have two species, each of which is present both in the bulk and on the surface. Further, suppose that each species can transfer between the surface and bulk via adsorption and desorption. Such situations arise, for example, when studying surfactants (21) or cell-level processes (22). We denote the concentrations of the species on the surface by u and v and the corresponding concentrations in the bulk by U and V. A possible model describing this type of behavior is:

$$\begin{aligned} u_t &= f(u,v) + D_u \nabla_S^2 u - \alpha_1 u + \beta_1 U, \text{ on } \mathcal{S}, \\ v_t &= g(u,v) + D_v \nabla_S^2 v - \alpha_2 v + \beta_2 V, \text{ on } \mathcal{S}, \\ U_t &= f(U,V) + D_U \nabla^2 U, \text{ in } \Omega, \\ V_t &= g(U,V) + D_V \nabla^2 V, \text{ in } \Omega, \end{aligned} \qquad [6]$$

with coupling boundary conditions:

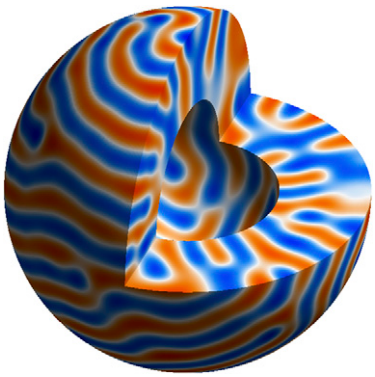

Fig. 3. A Turing pattern on a point cloud representing the 3D region between spheres of radii 0.5 and 1 (~800,000 randomly placed points form the point cloud). To visualize the result, we have cut away one-fourth of the region.

$$D_U \frac{\partial U}{\partial n} = \alpha_1 u - \beta_1 U, \quad D_V \frac{\partial V}{\partial n} = \alpha_2 v - \beta_2 V, \quad \text{on } \mathcal{S}. \qquad [7]$$

Here $\frac{\partial}{\partial n}$ represents the normal derivative outward from the bulk at the surface. The adsorption and desorption between the surface and bulk is controlled by the α and β parameters.

Fig. 5 shows an example computation where the surface is the *Drosophila* embryo from before but now coupled to an inner bulk process. Portions of the surface have been cut away in the visualization to show the bulk solution in the interior. Here the dynamics are the Schnakenberg process (23) for spot formation: $f(u,v) = \gamma(a - u + u^2 v)$ and $g(u,v) = \gamma(b - u^2 v)$. For parameter values, we use $\gamma = 30$, $a = 1/10$, $b = 9/10$, $D_u = D_U = 1/12$, $D_v = D_V = 1$, $\alpha_1 = \beta_1 = 5/12$, and $\alpha_2 = \beta_2 = 5$. The coupled *Drosophila* system is embedded inside a $5 \times 2.5 \times 2.5$ box that we discretize using a $240 \times 120 \times 120$ grid. Note that because the surface process is implemented on a regular Cartesian grid in 3D, we reuse this grid for the bulk process.

As in all of our examples, the numerical computation consists of alternating between two steps. The second step simply evolves u, v, U and V for one time-step using the same convolution-based technique as our previous examples. The first step extends the various quantities throughout the computational domain. For the quantities u and v, this extension is simply $u(\mathrm{cp}_{\mathcal{C}})$ and $v(\mathrm{cp}_{\mathcal{C}})$ as before. For the first two equations in Eq. **6**, we also need the value of the bulk quantities evaluated on the surface, but extended to the computational domain; these are $U(\mathrm{cp}_{\mathcal{C}})$ and $V(\mathrm{cp}_{\mathcal{C}})$. For the latter two equations in Eq. **6**, no extension is required for grid points in the interior of the *Drosophila* embryo. For grid points outside the *Drosophila* embryo, the values must be extended in such a way as to satisfy the coupling boundary conditions (Eq. **7**). Using the extensions already computed above, we approximate the surface values and the normal derivative as follows:

$$D_U \frac{U(\boldsymbol{x}) - U(\mathrm{cp}_{\mathcal{C}})}{\|\boldsymbol{x} - \mathrm{cp}_{\mathcal{C}}\|_2} = \alpha_1 u(\mathrm{cp}_{\mathcal{C}}) - \beta_1 U(\mathrm{cp}_{\mathcal{C}}),$$

where we have approximated $\frac{\partial U}{\partial n}$ using a finite difference in the normal direction. We then rearrange this formula for $U(\boldsymbol{x})$ to define our extension for U when $\boldsymbol{x}$ is an exterior point (U is left unchanged for interior points). This process is essentially an extrapolation based on the value of the normal derivative given on the surface to points off the surface. Unlike in the previous examples, this model requires knowledge of which grid points are inside of the surface and which are outside. For a simple surface such as the *Drosophila* embryo, we determine the interior by comparing the grid point $\boldsymbol{x}$, its closest point $\mathrm{cp}_{\mathcal{C}}(\boldsymbol{x})$, and the centroid of the point cloud. More sophisticated procedures to determine orientation may be necessary for complex geometry.

More generally, there could be bulk concentrations both inside and outside the surface. In such models, knowledge of the orientation of the surface is crucial to correctly approximate the coupling boundary conditions. Our approach to such problems assigns an orientation by splitting a bulk concentration U into two (global) quantities: U_1 (the inner concentration) and U_2 (the outer concentration).

Fig. 4. Fitzhugh–Nagumo waves of excitation (dark green) computed on the surface of the iconic heart (defined by 57,346 points).

Discussion

We now contrast the present approach with the other methods that have been developed for solving PDEs on surfaces. While most of these other methods do not immediately extend to point clouds, they could generally be used if the point cloud were first reconstructed into a suitable surface representation, a process that can be highly complex.

The first, and potentially most efficient, broad class of numerical methods for solving PDEs on surfaces are parameterization methods. These methods explicitly parameterize the surface by a smooth function of two Cartesian coordinates, α and β. Writing out the PDEs in terms of α- and β-derivatives and metric coefficients results in complicated formulas and, typically, singularities that require special care (24). To handle complex geometries and avoid such singularities, local patch approximations, such as splines, must be constructed that cover the entire surface as a collection of local smooth coordinate systems. This results in substantial organizational complexity. Note, however, there are recent methods that work on point clouds by performing the parameterization locally for each point in the cloud, using the nearest neighbors and a moving least squares problem (1). The second broad class of methods for solving PDEs on complex surfaces is to approximate the surface by a polyhedral triangulation, and use a finite element or finite volume type of method to discretize the equations (25, 26). In the case where only a point cloud is provided, these methods require an intermediate step of constructing a numerically well-behaved triangulation of the point cloud, which can be a considerable challenge for complex surfaces. Furthermore, discretizing and solving PDEs on such polyhedral grids can be deceptively difficult to implement, as described in ref. 26. The third and final broad class of methods are the embedding methods, which solve the surface PDEs in some 3D region encompassing the surface, using standard Cartesian-grid methods (3, 27–32). These methods do not use the metric tensor on the surface, thus eliminating one of the major technical complexities of working with PDEs on surfaces. However, embedding methods typically alter the surface PDEs and introduce artificial boundary conditions in sophisticated ways to extend the problem to a 3D neighborhood of the surface. Traditional embedding methods also require some continuous surface representation, typically the zero-level set of a smooth function defined on all of space (as in the level set method), rather than a point cloud. The accurate and efficient construction of such representations of the surface is a challenge

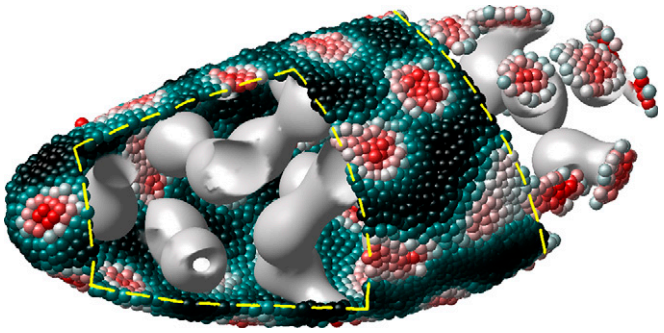

Fig. 5. Coupling a diffusion equation in the bulk to a Turing surface pattern formation problem. Some of the point cloud $\mathcal{C}$ has been hidden to see the interior bulk.

that remains a subject of current research interest, especially for open surfaces, or surfaces of codimension-two or higher (33, 34).

The method presented here, the closest point method, belongs to the general class of embedding methods, but distinguishes itself in several major respects. In particular, the extended PDE for the closest point method is just the classical 3D analog of the surface PDE, a fact that enables the use of standard 3D discretizations. Also, the representation of the surface (i.e., the closest point function) is simple and robust, and it immediately extends beyond surfaces to arbitrary point clouds, and thus directly to the input data format. In the special case of pure diffusion, the present method bears some similarity to the embedding method introduced by Schwartz et al. (27). In particular, both methods rely on the fact that under suitable conditions the solution to a standard 3D diffusion equation provides a close approximation to the solution of the surface diffusion equation. However, ref. 27 formulates the problem using "cut cell" discretizations, the introduction of Neumann boundary conditions, and sophisticated level-set PDE methods for reconstructing the surface of interest as a level set function. In contrast, the closest point method uses only a standard Cartesian grid discretization, no boundary condition discretizations, and is based on the easily computed closest point function. The last of these properties ensures that no surface reconstruction is needed, and allows the method to be immediately generalized from surfaces to point clouds, and other more general geometric objects (e.g., objects of mixed dimensions).

As we have seen, our method solves systems of reaction–diffusion equations on surfaces. The surface representation can be given as an unorganized point set, a property that makes the method easy to implement for complex surfaces such as those acquired from laser range scanners. Note, however, that the fundamental principles underlying the closest point method apply to much more general PDEs. See, for example, refs. 3, 5, 35, where the closest point method is used to solve an interesting variety of PDEs on complex continuous surfaces. Our ongoing work for the closest point method investigates solving general PDEs on general point cloud surfaces using only classical 3D numerical methods and solution routines. We anticipate that the development of methods of this type should encourage powerful models of continuum processes on surfaces to be applied more widely, across diverse sciences and in new research areas.

ACKNOWLEDGMENTS. C.B.M. thanks Dr. Chandrasekhar Venkataraman (University of Sussex) for useful discussions on bulk-coupled reaction–diffusion models. The work of C.B.M. was supported by Award KUK-C1-013-04 from King Abdullah University of Science and Technology (KAUST). The work of S.J.R. was partially supported by a Natural Sciences and Engineering Research Council of Canada Discovery Grant and by Award KUK-C1-013-04 from KAUST.

1. Liang J, Zhao H-K (2012) Solving partial differential equations on point clouds. *UCLA CAM Reports* 12–25.
2. Lai R-J, Liang J, Zhao H-K (2012) A local mesh method for solving PDEs on point clouds. *UCLA CAM Report* 12–60.
3. Ruuth SJ, Merriman B (2008) A simple embedding method for solving partial differential equations on surfaces. *J Comput Phys* 227(3):1943–1961.
4. März T, Macdonald CB (2012) Calculus on surfaces with general closest point functions. *SIAM J Numer Anal* 50(6):3303–3328.
5. Macdonald CB, Ruuth SJ (2009) The implicit closest point method for the numerical solution of partial differential equations on surfaces. *SIAM J Sci Comput* 31(6):4330–4350.
6. Boyd JP (2001) *Chebyshev and Fourier Spectral Methods,* 2nd Rev Ed (Courier Dover Publications, New York).
7. Spivak M, Veerapaneni SK, Greengard L (2010) The fast generalized Gauss transform. *SIAM J Sci Comput* 32(5):3092–3107.
8. Greengard L, Strain J (1991) The fast Gauss transform. *SIAM J Sci Statist Comput* 12(1): 79–94.
9. Saff EB, Kuijlaars ABJ (1997) Distributing many points on a sphere. *The Mathematical Intelligencer* 19(1):5–11.
10. Pearson JE (1993) Complex patterns in a simple system. *Science* 261(5118):189–192.
11. Turing AM (1952) The chemical basis of morphogenesis. *Philos Trans R Soc Lond* B237 (641):37–72.
12. Berkeley Drosophila Transcription Network Project (2010) http://bdtnp.lbl.gov. Accessed January 1, 2010.
13. Prigogine I, Lefever R (1968) Symmetry breaking instabilities in dissipative systems. II. *J Chem Phys* 48(4):1695–1700.
14. NextEngine (2009) *NextEngine "shell".* www.nextengine.com/gallery/shell. Accessed May 2, 2009.
15. Kobbelt L, Botsch M (2004) A survey of point-based techniques in computer graphics. *Comput Graph* 28(6):801–814.
16. Auer A, Macdonald CB, Treib M, Schneider J, Westermann R (2012) Real-time fluid effects on surfaces using the Closest Point Method. *Comput Graph Forum* 31(6):1909–1923.
17. Fitzhugh R (1961) Impulses and physiological states in theoretical models of nerve membrane. *Biophys J* 1(6):445–466.
18. Nagumo J, Arimoto S, Yoshizawa S (1962) Active pulse transmission line simulating nerve axon. *Proc IRE* 50(10):2061–2070.
19. Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol* 117(4):500–544.
20. Cbspicer (2008) *"Iconic Heart" from the TurboSquid 3D stock catalog.* www.turbosquid.com/FullPreview/Index.cfm/ID/389728. Accessed January 20, 2011.
21. Teigen KE, Li X, Lowengrub J, Wang F, Voigt A (2009) A diffuse-interface approach for modeling transport, diffusion and adsorption/desorption of material quantities on a deformable interface. *Commun Math Sci* 4(7):1009–1037.
22. Novak IL, et al. (2007) Diffusion on a curved surface coupled to diffusion in the volume: Application to cell biology. *J Comput Phys* 226(2):1271–1290.
23. Schnakenberg J (1979) Simple chemical reaction systems with limit cycle behaviour. *J Theor Biol* 81(3):389–400.
24. Floater MS, Hormann K (2005) Surface parametrization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling,* eds Dodgson N, Floater M, Sabin M (Springer, Heidelberg, Germany), pp 157–186.
25. Dziuk G, Elliott CM (2008) Eulerian finite element method for parabolic PDEs on implicit surfaces. *Interf Free Bound* 10:119–138.
26. Reuter M, Wolter FE, Peinecke N (2006) Laplace-Beltrami spectra as 'Shape-DNA' of surfaces and solids. *Comput Aided Des* 38(4):342–366.
27. Schwartz P, Adalsteinsson D, Colella P, Arkin AP, Onsum M (2005) Numerical computation of diffusion on a surface. *Proc Natl Acad Sci USA* 102(32):11151–11156.
28. Bertalmio M, Cheng L-T, Osher S, Sapiro G (2001) Variational problems and partial differential equations on implicit surfaces. *J Comput Phys* 174(2):759–780.
29. Xu J-J, Zhao H-K (2003) An Eulerian formulation for solving partial differential equations along a moving interface. *J Sci Comput* 19(1-3):573–594.
30. Leung S-Y, Zhao H-K (2009) A grid based particle method for moving interface problems. *J Comput Phys* 228(8):2993–3024.
31. Leung S-Y, Lowengrub J, Zhao H-K (2011) A grid based particle method for solving partial differential equations on evolving surfaces and modeling high order geometrical motion. *J Comput Phys* 230(7):2540–2561.
32. Greer JB (2006) An improvement of a recent Eulerian method for solving PDEs on general geometries. *J Sci Comput* 29(3):321–352.
33. Zhao H-K, Osher S, Fedkiw R (2001) Fast surface reconstruction using the level set method. In *Proceedings of the IEEE Workshop on Variational and Level Set Methods* (IEEE Computer Society, Vancouver, Canada), 194–201.
34. Goldstein T, Bresson X, Osher S (2010) Geometric applications of the split Bregman method: segmentation and surface reconstruction. *J Sci Comput* 45(1-3):272–293.
35. Macdonald CB, Ruuth SJ (2008) Level set equations on surfaces via the Closest Point Method. *J Sci Comput* 35(2-3):219–240.