Diffusion Equation in Irregular Domains

Moo K. Chung

University of Wisconsin-Madison, USA

mkchung@wisc.edu

In brain imaging, the image acquisition and processing processes themselves are likely to introduce noise to the images. It is therefore imperative to reduce the noise while preserving the geometric details of the anatomical structures for various applications. Traditionally Gaussian kernel smoothing has been often used in brain image processing and analysis. However, the direct application of Gaussian kernel smoothing tend to cause various numerical issues in irregular domains with boundaries. For example, if one uses large bandwidth in kernel smoothing in a cortical bounded region, the smoothing will blur signals across boundaries. So in kernel smoothing and regression literature, various ad-hoc procedures were introduce to remedy the boundary effect. We show how to solve diffusion equations in irregular domains using the polynomial approximation.

1 Introduction

Motivated by Perona & Malik (1990), diffusion equations have been widely used in brain imaging as a form of noise reduction. The most natural straightforward way to smooth images in irregular domains with boundaries is to formulate the problem as boundary value problems using partial differential equations. Numerous diffusion-based techniques have been developed in image processing (Sochen et al. 1998, Malladi & Ravve 2002, Tang et al. 1999, Taubin 2000, Andrade et al. 2001, Chung et al. 2001, Chung, Worsley, Robbins, Paus, Taylor, Giedd, Rapoport & Evans 2003, Chung et al. 2005, Chung & Taylor 2004, Cachia, Mangin, Riviére, Papadopoulos-Orfanos, Kherif, Bloch & Régis 2003, Cachia, Mangin, Riviére, Kherif, Boddaert, Andrade, Papadopoulos-Orfanos, Poline, Bloch, Zilbovicius, Sonigo, Brunelle & Régis 2003, Joshi et al. 2009). In this paper, we will overview the basics of isotropic diffusion equations and explain how to solve them on regular grids and irregular grids such as graphs.

2 Finite difference method

One way of solving diffusion equations numerically in to use finite differences. We will discuss how to differentiate images. There are numerous techniques for differentiation proposed in literature. We start with simple example of image differentiation in 2D image slices. Consider image intensity $f(x, y)$ defined on a regular grid, i.e., $(x, y) \in \mathbb{Z}^2$. Assume the pixel size is δx and δy in x- and y-directions. The partial derivative along the x-direction of image f is approximated by the finite difference:

$$\frac{\partial f}{\partial x}(x,y) = \frac{f(x+\delta x, y) - f(x,y)}{\delta x}.$$

The partial derivative along the y-direction of image f is approximated similarly. $\frac{\partial f}{\partial x}(x,y)$ and $\frac{\partial f}{\partial y}(x,y)$ are called the *first order derivatives*. Then the *second order derivatives* are defined by taking the finite difference twice:

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2}(x,y) &= \Big[\frac{f(x+\delta x, y) - f(x,y)}{\delta x} - \frac{f(x,y) - f(x-\delta x, y)}{\delta x}\Big]/\delta x \\ &= \frac{f(x+\delta x, y) - 2f(x,y) + f(x-\delta x, y)}{\delta x^2}\end{aligned}$$

Similarly, we also have

$$\frac{\partial^2 f}{\partial y^2}(x,y) = \frac{f(x, y+\delta y) - 2f(x,y) + f(x, y-\delta y)}{\delta x^2}.$$

Other partial derivatives such as $\frac{\partial^2 f}{\partial x \partial y}$ are computed similarly.

3 1D diffusion by finite difference

Let us implement 1D version of diffusion equations (Figure 1). Suppose we have a smooth function $f(x,t)$ which is a function of position $x \in \mathbb{R}$ and time $t \in \mathbb{R}^+$. 1D isotropic heat equation is then defined as

$$\frac{\partial f}{\partial t} = \frac{d^2 f}{dx^2} \tag{1}$$

with initial condition $f(x, t=0) = g(x)$. Differential equation (1) is then discretized as

$$f(x, t+\delta t) = f(x,t) + \delta t \frac{d^2 f}{dx^2}(x,y). \tag{2}$$

With $t_k = k\delta t$ and starting from $t = 0$, (2) can be written as

$$f(x, t_{k+1}) = f(x, t_k) + \delta t \frac{f(x+\delta x, t_k) - 2f(x,t_k) + f(x-\delta x, t_k)}{\delta x^2}. \tag{3}$$

The above finite difference gives the solution at time t_{k+1}. To obtain the solution at any time, it is necessary to keep iterating many times with very small δt. If δt is too small, the computation is slow. If it is too large, the finite difference will diverge. Then the problem is finding the largest δt that grantee the convergence.

Numerically (3) is solved in MATLAB follows. We start with generating a step function as the ground truth (black line in Figure 2). We then add $N(0, 0.5^2)$ noise.

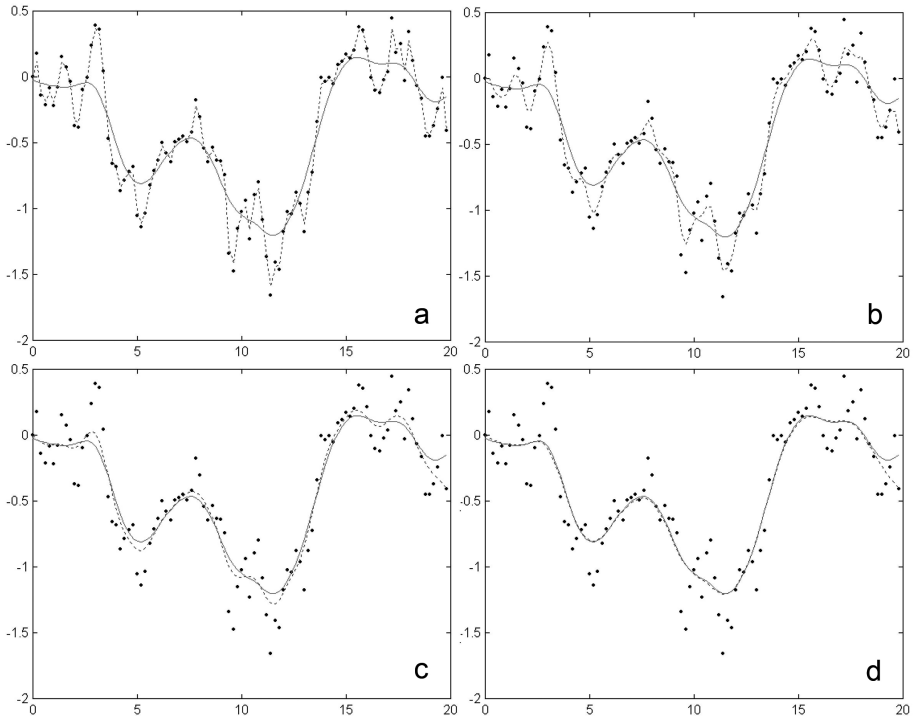

Fig. 1: Gaussian kernel smoothing (solid line) and diffusion smoothing (dotted line). (a) before diffusion (b) after 0.05 seconds (5 iterations) (c) after 0.25 seconds (25 iterations) (d) after 0.5 seconds (50 iterations).

```
x=1:1000;
noise=normrnd(0, 0.5, 1,1000);
signal= [zeros(1,300) ones(1,400) zeros(1,300)];
figure; plot(signal, 'k', 'LineWidth',2);
y= signal + noise;
hold on; plot(y, ':k');
```

The 2nd order finite difference is coded as `L=[1 -2 1]`. Then it is convoluted with 3 consecutive data at a time in the code blow.

```
L = [1 -2 1]
g=y;
for i=1:10000
    Lg = conv(g,L,'same');
    g = g+ 0.01*Lg;
end
hold on;plot(g, 'b', 'LineWidth', 2);
```

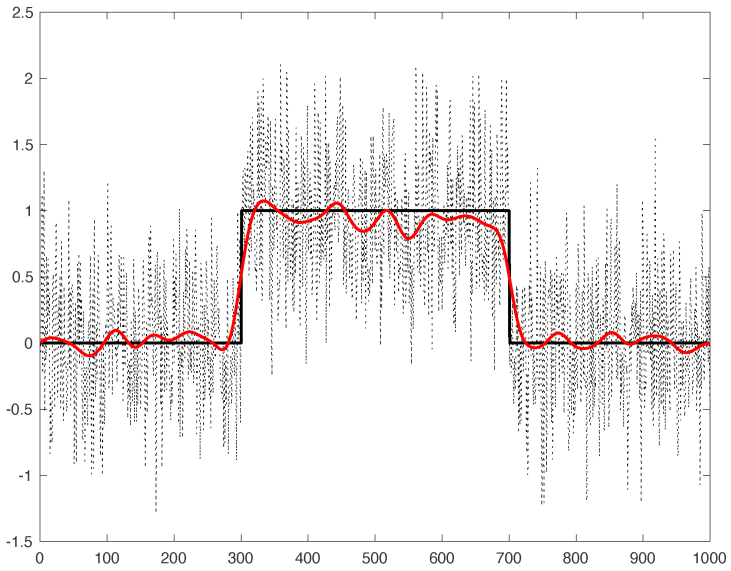

Fig. 2: Diffusion of simulated data (dotted line) and the ground truth (black line). The both methods using `conv.m` and `toeplitz.m` all converge to the red line.

Since the Laplacian is a linear operator, the above convolution can be written as the matrix multiplication. Any linear operation can be discretely encoded as matrix multiplication. Here the Laplacian is encoded using a Toeplitz matrix:

```
c=zeros(1,1000);
c(1:2)=[-2 1];
r=zeros(1,1000);
r(1:2)=[-2 1];
L = toeplitz(c,r);
```

The first 5 columns and rows of the Toelitz matrix `L` is given by

```
 L(1:5,1:5)
 ans =
     -2     1     0     0     0
      1    -2     1     0     0
      0     1    -2     1     0
      0     0     1    -2     1
      0     0     0     1    -2
```

The diffusion is solved by sequential summation of matrix multiplications

```
g=y';
for i=1:10000
```

```
    g = g+ 0.01*L*g;
end
hold on;plot(g, 'g', 'LineWidth', 2);
```

The first and last rows of the the Toelitz matrix `L` is `[-2 1 0]` and `[0 1 -2]`, which is different from the 2nd order finite difference `[1 -2 1]`. It will not matter since that may be viewed as the discrete Laplacian in the boundary. The matrix form of Laplacian can be used in conjunction with the recently developed polynomial approximation method for solving heat diffusion on manifolds (Huang et al. 2020).

Discrete maximum principle. Since the diffusion smoothing and kernel smoothing are equivalent, The diffused signal $f(x, t_{k+1})$ must be bounded by the minimum and the maximum of signal (Chung, Worsley, Robbins & Evans 2003). Let x_{i-1}, x_i, x_{i+1} be some points with gap δx.

$$\begin{aligned} f(x_i, t_{k+1}) &= f(x_i, t_k) + \delta t \frac{d^2 f}{dx^2}(x_i, t_k) \\ &\leq \max\big[f(x_{i-1}, t_k), f(x_i, t_k), f(x_{i+1}, t_k)\big]. \end{aligned}$$

Similarly, we can bound it below. Thus, the time step should be bounded by

$$\delta t \leq \max\Big[\Big|\frac{f(x_{i-1}, t_j) - f(x_i, t_j)}{\frac{d^2 f}{dx^2}}\Big|, \Big|\frac{f(x_{i+1}, t_j) - f(x_i, t_j)}{\frac{d^2 f}{dx^2}}\Big|\Big].$$

4 Diffusion in n-dimensional grid

In 2D, let (x_i, y_i) be pixels around (x, y) including (x, y) itself. Then using the 4-neighbor scheme, Laplcian of $f(x, y)$ can be written as

$$\Delta f(x, y) = \sum_{i,j} w_{ij} f(x_i, y_i),$$

where the Laplacian matrix is given by

$$(w_{ij}) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Note that $\sum_{ij} w_{ij} = 1$.

Extending it further, consider nD. Let $x = (x_1, x_2, \cdots, x_n)$ be the coordinates in $\mathbb{R}^n$. Laplacian Δ in $\mathbb{R}^n$ is defined as

$$\Delta f = \frac{\partial^2 f}{\partial x_1^2} + \cdots + \frac{\partial^2 f}{\partial x_n^2}.$$

Assume we have a n-dimensional hyper-cube grid of size is 1. Then we have

$$\begin{aligned}\Delta f(x) = & f(x_1 \pm 1, \cdots, x_n) + \cdots + f(x_1, \cdots, x_n \pm 1) \\ & -2nf(x, y).\end{aligned}$$

This uses $2n$ closest neighbors of voxel x to approximate the Laplacian.

It is also possible to incorporate 2^n corners $(x_1 \pm 1, \cdots, x_n \pm 1)$ along with the $2n$ closest neighbors for a better approximation of the Laplacian. In particular in 2D, we can obtain a more accurate finite difference formula for 8-neighbor Laplacian:

$$(w_{ij}) = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Based on the estimation Laplacian on discrete grid, diffusion equation

$$\frac{\partial f}{\partial t} = \Delta f$$

is discretized as

$$f(x, t_{k+1}) = f(x, t_k) + \delta t \sum_{i,j} w_{ij} f(x_i, y_i). \tag{4}$$

with $t_k = k\delta t$ and starting from $t_1 = 0$. From (4), we can see that the diffusion equation is solved by iteratively applying convolution with weights w_{ij}. In fact, it can be shown that the solution of diffusion is given by kernel smoothing.

5 Laplacian on Irregular Domains

Now we generalize volumetric Laplacian in previous sections to graphs. Let $G = (V, E)$ be a graph with node set V and edge set E. We will simply index the node set as $V = \{1, 2, \cdots, p\}$. If two nodes i and j form an edge, we denote it as $i \sim j$. Let $W = (w_{ij})$ be the edge wight. The adjacency matrix of G is often used as the edge weight. Various forms of graph Laplacian have been proposed (Chung & Yau 1997) but the most often used standard form $L = (l_{ij})$ is given by

$$l_{ij} = \begin{pmatrix} -w_{ij}, & i \sim j \\ \sum_{i \neq j} w_{ij}, & i = j \\ 0, & \text{otherwise} \end{pmatrix}$$

Often it is defined with the sign reversed such that

$$l_{ij} = \begin{pmatrix} w_{ij}, & i \sim j \\ -\sum_{i \neq j} w_{ij}, & i = j \\ 0, & \text{otherwise} \end{pmatrix}$$

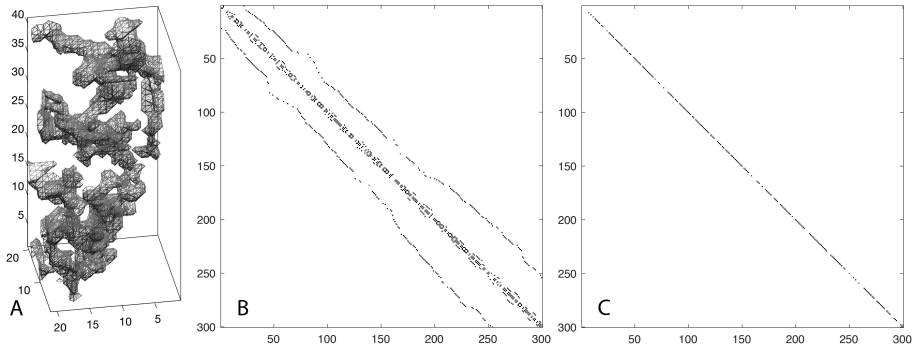

Fig. 3: A. Part of lung blood vessel obtained from CT. B. Adjacency matrix obtained from 4-neighbor connectivity. C. Laplace matrix obtained from the adjacency matrix.

The graph Laplacian L can then be written as

$$L = D - W,$$

where $D = (d_{ij})$ is the diagonal matrix with $d_{ii} = \sum_{j=1}^{n} w_{ij}$. Here, we will simply use the adjacency matrix so that the edge weights w_{ij} are either 0 or 1. In Matlab, Laplacian `L` is simply computed from the adjacency matrix `adj`:

```
n=size(adj,1);
adjsparse = sparse(n,n);
adjsparse(find(adj))=1;
L=sparse(n,n);
GL = inline('diag(sum(W))-W');
L = GL(adjsparse);
```

We use the sparse matrix format to reduce the memory burden for large-scale computation.

Theorem 1. *Graph Laplacian L is nonnegative definite.*

The proof is based on factoring Laplacian L using incidence matrix ∇ such that $L = \nabla^\top \nabla$. Such factorization always yields nonnegative definite matrices. Very often L is nonnegative definite in practice if it is too sparse (Figure 3).

Theorem 2. *For graph Laplacian L, $L + \alpha I$ is positive definite for any $\alpha > 0$.*

Proof. Since L is nonnegative definite, we have

$$x^\top L x \geq 0.$$

Then it follows that

$$x^\top (L + \alpha I) x = x^\top L x + \alpha x^\top x > 0$$

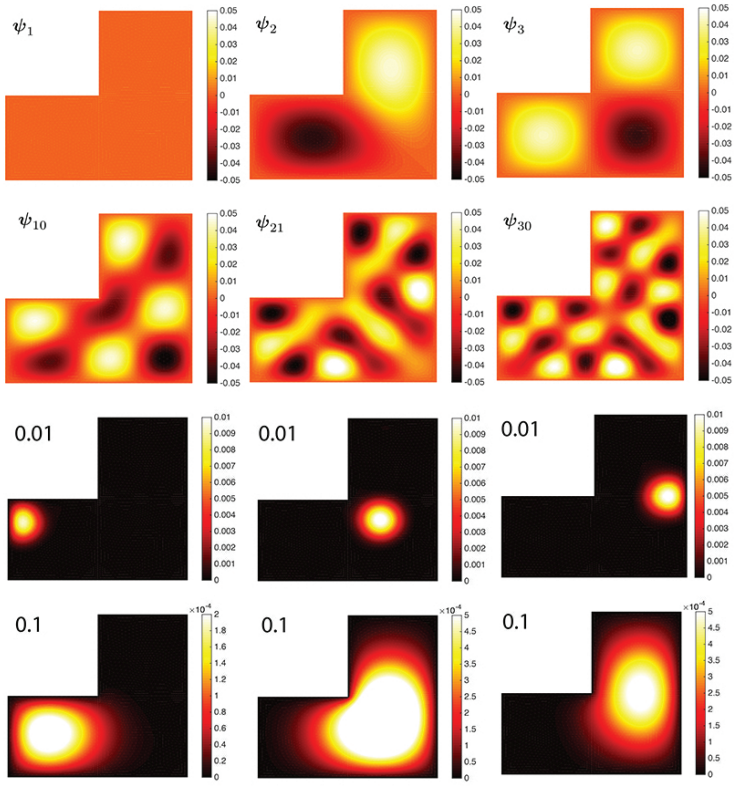

Fig. 4: Top: First few eigenvectors of the Laplacian in a L-shaped domain. Botom: Heat kernel with bandwidths $\sigma = 0.01, 0.1$. We have used degree 70 expansions but the shape is almost identical if we use higher degree expansions. The heat kernel is a probability distribution that follows the shape of the L-shaped domain.

for any $\alpha > 0$ and $x \neq 0$. □

Unlike the continuous Laplace-Beltrami operators that may have possibly infinite number of eigenfunctions, we have up to p number of eigenvectors $\psi_1, \psi_2, \cdots, \psi_p$ satisfying

$$L\psi_j = \lambda_j \psi_j \tag{5}$$

with (Figure 4)

$$0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_p.$$

The eigenvectors are orthonormal, i.e.,

$$\psi_i^\top \psi_j = \delta_{ij},$$

the Kroneker's delta. The first eigenvector is trivially given as $\psi_1 = \mathbf{1}/\sqrt{p}$ with $\mathbf{1} = (1, 1, \cdots, 1)^\top$.

All other higher order eigenvalues and eigenvectors are unknown analytically and have to be computed numerically (Figure 4). Using the eigenvalues and eigenvectors, the graph Laplacian can be decomposed spectrally. From (5),

$$L\Psi = \Psi\Lambda, \tag{6}$$

where $\Psi = [\psi_1, \cdots, \psi_p]$ and Λ is the diagonal matrix with entries $\lambda_1, \cdots, \lambda_p$. Since Ψ is an orthogonal matrix,

$$\Psi\Psi^\top = \Psi^\top\Psi = \sum_{j=1}^{p} \psi_j\psi_j^\top = I_p,$$

the identify matrix of size p. Then (6) is written as

$$L = \Psi\Lambda\Psi^\top = \sum_{j=1}^{p} \lambda_j\psi_j\psi_j^\top.$$

This is the restatement of the singular value decomposition (SVD) for Laplacian.

For measurement vector $f = (f_1, \cdots, f_p)^\top$ observed at the p nodes, the discrete Fourier series expansion is given by

$$f = \sum_{j=1}^{n} \tilde{f}_j\psi_j,$$

where $\tilde{f}_j = f^\top\psi_j = \psi_j^\top f$ are Fourier coefficients.

6 Polynomial Approximation

In performing diffusion on scalar functional data defined at the vertices of a triangulated mesh, significant computational reduction can be achieved by approximating the solution of the heat diffusion equation using a polynomial expansion of the graph Laplacian Δ (Huang et al. 2019). The diffusion on graphs or triangle mesh is expressed as

$$\frac{\partial u}{\partial t} = \Delta u,$$

where u represents the scalar function undergoing diffusion, t is the time variable, and Δ is the Laplacian.

The heat diffusion can be represented as a convolution of the initial data u_0 with the heat kernel $K_t = e^{-\Delta t}$:

$$u(p,t) = \int_{\mathcal{M}} K_t(p,q)u_0(q)\, d\mu(q),$$

where $d\mu(q)$ is the measure over $\mathcal{M}$. The solution to heat diffusion is then given in the matrix form as

$$u(t) = e^{-\Delta t}u_0.$$

We expand the heat kernel $e^{-\Delta t}$ using the series of orthogonal polynomials as

$$e^{-\Delta t} = \sum_{n=0}^{\infty} c_{t,n} P_n(\Delta),$$

where $P_n(\lambda)$ are orthogonal polynomials, and $c_{t,n}$ are the expansion coefficients. For Laguerre polynomials $P_n(\lambda)$, these coefficients are explicitly given by:

$$c_{t,n} = \frac{t^n}{(t+1)^{n+1}}.$$

Using this expansion, the diffusion process at time t is given by

$$u(p,t) = K_t * u = \sum_{n=0}^{\infty} c_{t,n} P_n(\Delta) u_0,$$

This polynomial approximation provides an efficient means of computing heat diffusion without explicitly computing the spectral components of the Laplacian.

To avoid the explicit computation of $P_n(\Delta)u_0$, which can be computationally expensive, the polynomial terms $P_n(\Delta)u_0$ are computed using the recurrence relation:

$$P_{n+1}(\Delta)u_0 = (\alpha_n \Delta + \beta_n) P_n(\Delta)u_0 + \gamma_n P_{n-1}(\Delta)u_0,$$

with initial conditions $P_{-1}(\Delta)u_0 = 0$ and $P_0(\Delta_0)u_0 = u_)$. For Laguerre polynomials, the recurrence coefficients are:

$$\alpha_n = -\frac{1}{n+1}, \quad \beta_n = \frac{2n+1}{n+1}, \quad \gamma_n = -\frac{n}{n+1}.$$

This approach enables efficient computation of diffusion.

7 Laplace equation

In this section, we will show how to solve for steady state of diffusion. The distribution of fictional charges within the two boundaries sets up a scalar potential field f, which satisfies the Poisson equation

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = \frac{\rho}{\epsilon_0},$$

where ρ is the total charge within the boundaries. If we set up the two boundaries at different potential, say at f_0 and f_1, without enclosing any charge, we have the Laplace equation

$$\Delta f = 0.$$

The Laplace equation can be viewed as the steady state of diffusion

$$\frac{df(t,p)}{dt} = \Delta f$$

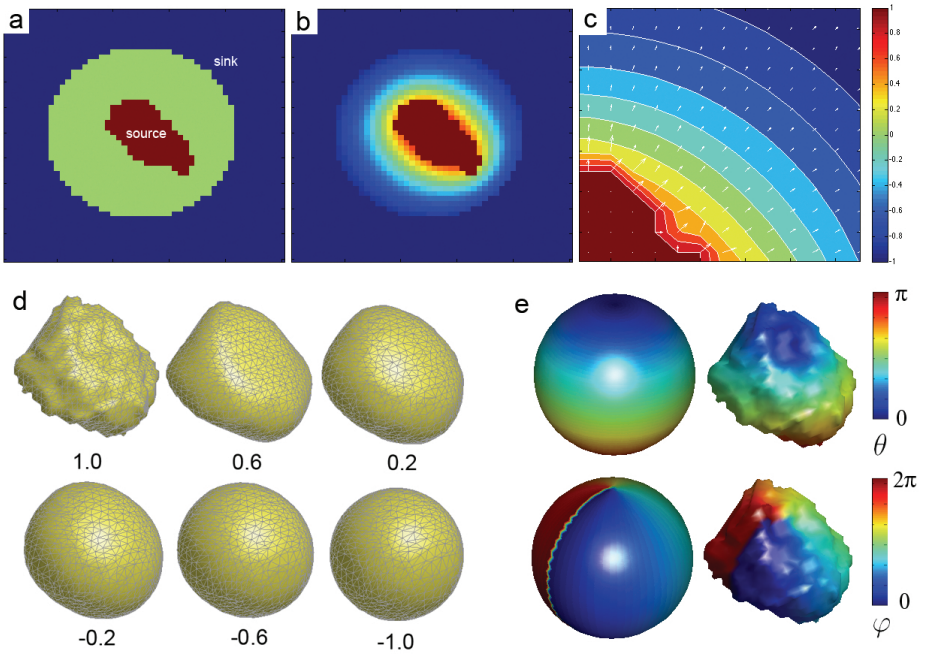

Fig. 5: (a) The heat source (amygdala) is assigned value 1 while the heat sink (outer sphere) is assigned the value -1. The diffusion equation is solved with these boundary conditions. (b) After a sufficient number of iterations, the equilibrium state $f(\infty, p)$ is reached. (c) The gradient field $\nabla f(t = \infty, p)$ shows the direction of heat propagation from the source to the sink. The integral curve of the gradient field is computed by connecting one level set to the next level set of $f(\infty, p)$. (d) The deformation of amygala surface to the sphere is done by tracing the integral curve at each mesh vertex. The numbers $c = 1.0, 0.6, \cdots, -1.0$ correspond to the level sets $f(\infty, p)$. (e) The surface flattening to the sphere produces the surface parameterization based on the spherical angles (θ, φ). The point $\theta = 0$ corresponds to the north pole of a unit sphere. The method presented here is published in (Chung et al. 2010, Chung 2012).

when $t \to \infty$. By solving the Laplace equation with the two boundary conditions, we obtain the potential field f. Then the electric field perpendicular to the isopotential surfaces is given by $-\nabla f$. The Laplace equation is mainly solved using the finite difference scheme (Chung 2012). The electric field lines radiate from one conducting surface to the other without crossing each other. By tracing the electric field lines, we obtain one-to-one smooth map between surfaces (Figure 5). The underlying framework is identical to the Laplace equation based surface flattening or cortical thickness estimation (Jones et al. 2000, Chung et al. 2010).

Bibliography

Andrade, A., Kherif, F., Mangin, J., Worsley, K., Paradis, A., Simon, O., Dehaene, S., Le Bihan, D. & Poline, J.-B. (2001), 'Detection of fMRI activation using cortical surface mapping', *Human Brain Mapping* **12**, 79–93.

Cachia, A., Mangin, J.-F., Riviére, D., Kherif, F., Boddaert, N., Andrade, A., Papadopoulos-Orfanos, D., Poline, J.-B., Bloch, I., Zilbovicius, M., Sonigo, P., Brunelle, F. & Régis, J. (2003), 'A primal sketch of the cortex mean curvature: A morphogenesis based approach to study the variability of the folding patterns', *IEEE Transactions on Medical Imaging* **22**, 754–765.

Cachia, A., Mangin, J.-F., Riviére, D., Papadopoulos-Orfanos, D., Kherif, F., Bloch, I. & Régis, J. (2003), 'A generic framework for parcellation of the cortical surface into gyri using geodesic Voronoï diagrams', *Image Analysis* **7**, 403–416.

Chung, F. & Yau, S. (1997), 'Eigenvalue inequalities for graphs and convex subgraphs', *Communications in Analysis and Geometry* **5**, 575–624.

Chung, M. (2012), *Computational Neuroanatomy: The Methods*, World Scientific, Singapore.

Chung, M., Robbins, S. & Evans, A. (2005), 'Unified statistical approach to cortical thickness analysis', *Information Processing in Medical Imaging (IPMI), Lecture Notes in Computer Science* **3565**, 627–638.

Chung, M. & Taylor, J. (2004), Diffusion smoothing on brain surface via finite element method, *in* 'Proceedings of IEEE International Symposium on Biomedical Imaging (ISBI)', Vol. 1, pp. 432–435.

Chung, M., Worsley, K., Brendon, M., Dalton, K. & Davidson, R. (2010), 'General multivariate linear modeling of surface shapes using SurfStat', *NeuroImage* **53**, 491–505.

Chung, M., Worsley, K., Robbins, S. & Evans, A. (2003), Tensor-based brain surface modeling and analysis, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. I, pp. 467–473.

Chung, M., Worsley, K., Robbins, S., Paus, T., Taylor, J., Giedd, J., Rapoport, J. & Evans, A. (2003), 'Deformation-based surface morphometry applied to gray matter deformation', *NeuroImage* **18**, 198–213.

Chung, M., Worsley, K., Taylor, J., Ramsay, J., Robbins, S. & Evans, A. (2001), 'Diffusion smoothing on the cortical surface', *NeuroImage* **13**, S95.

Huang, S.-G., Lyu, I., Qiu, A. & Chung, M. (2019), 'Fast polynomial approximation to heat diffusion in manifolds', *MICCAI* **11767**, 48–56.

Huang, S.-G., Lyu, I., Qiu, A. & Chung, M. (2020), 'Fast polynomial approximation of heat kernel convolution on manifolds and its application to brain sulcal and gyral graph pattern analysis', *IEEE Transactions on Medical Imaging* **39**, 2201–2212.

Jones, S., Buchbinder, B. & Aharon, I. (2000), 'Three-dimensional mapping of cortical thickness using Laplace's equation', *Human Brain Mapping* **11**, 12–32.

Joshi, A., Shattuck, D. W., Thompson, P. M. & Leahy, R. M. (2009), 'A parameterization-based numerical method for isotropic and anisotropic diffusion smoothing on non-flat surfaces', *IEEE Transactions on Image Processing* **18**, 1358–1365.

Malladi, R. & Ravve, I. (2002), Fast difference schemes for edge enhancing Beltrami flow, *in* 'Proceedings of Computer Vision-ECCV, Lecture Notes in Computer Science (LNCS)', Vol. 2350, pp. 343–357.

Perona, P. & Malik, J. (1990), 'Scale-space and edge detection using anisotropic diffusion', *IEEE Trans. Pattern Analysis and Machine Intelligence* **12**, 629–639.

Sochen, N., Kimmel, R. & Malladi, R. (1998), 'A general framework for low level vision', *IEEE Transactions on Image Processing* **7**, 310–318.

Tang, B., Sapiro, G. & Caselles, V. (1999), Direction diffusion, *in* 'The Proceedings of the Seventh IEEE International Conference on Computer Vision', pp. 2:1245–1252.

Taubin, G. (2000), Geometric signal processing on polygonal meshes, *in* 'EUROGRAPHICS', Eurographics Association.