The Waisman Laboratory for Brain Imaging and Behavior

University of Wisconsin SCHOOL OF MEDICINE AND PUBLIC HEALTH

Diffusion Equations

Moo K. Chung
Department of Biostatistics and Medical Informatics
Waisman Laboratory for Brain Imaging and Behavior
University of Wisconsin-Madison

www.stat.wisc.edu/~mchung

Differential Equations in Euclidean space

Linear operators

Operation *f* is linear if

$$f(a p+b q)=a f(p)+b f(q)$$

Convolution, integration, differentiation are all linear operations.

Q: Is $y=x^2$ a linear function?

Image derivatives

Image intensity is assumed to be the discrete observation of a smooth continuous signal.

Its 1st order partial derivatives:

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

Take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}(x,y) = f(x+1, y) - f(x,y)$$

Exercise: Implement the derivatives as kernel convolutions.

$\frac{\partial f}{\partial x}$:

$\frac{\partial f}{\partial y}$:

MATLAB implementation

`Y = diff(X)`

calculates differences between adjacent elements of X along the first array dimension.

If X is a vector of length *m*, then Y = diff(X) returns a vector of length *m*-1. The elements of Y are the differences between adjacent elements of X.

```
Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]
```

`Y = diff(X,N,DIM)` is the Nth difference function along dimension DIM. For image I, the 2nd order derivatives are given by

```
diff(I,2,1) and diff(I,2,2)
```

First derivative as matrix multiplication

```
Y = diff(X)
Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]
```

$$
\begin{bmatrix} X(2)-X(1) \\ X(3)-X(2) \\ X(4)-X(3) \\ . \\ . \\ . \\ . \\ X(m)-X(m-1) \\ -X(m) \end{bmatrix}
=
\begin{bmatrix}
-1 & 1 & 0 & & & & 0 \\
0 & -1 & 1 & & & & . \\
0 & 0 & -1 & & & & . \\
0 & 0 & 0 & & & & \\
0 & 0 & 0 & & & & \\
. & . & . & & & & . \\
 & & & . & & & . \\
 & & & & & -1 & 1 \\
0 & & & . & . & 0 & -1
\end{bmatrix}
\times
\begin{bmatrix} X(1) \\ X(2) \\ X(3) \\ . \\ . \\ . \\ . \\ X(m-1) \\ X(m) \end{bmatrix}
$$

Toeplitz matrix

Functional data

Coding first derivative using Toeplitz matrix

```
-1        1        0             0
 0       -1        1             .
 0        0       -1             .
 0        0        0
 0        0        0
 .        .        .             .
                      .          .
                            -1   1
 0                    .  .   0  -1
```

```
c=zeros(1,T);                          %first
column
c(1)=-1;
r=zeros(1,T);                          %first row
r(1:2)=[-1 1];
L1 = toeplitz(c,r);                 %1st finite
```

2nd derivatives

The 1st order partial derivatives:

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

The 2nd order partial derivatives:

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

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

Exercise: Implement the 2nd order derivatives as kernel convolution.

Second derivatives

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

$$\frac{\partial^2 f}{\partial x^2}(x,y) =$$

$$\frac{\partial f}{\partial x}:$$

$$\frac{\partial f}{\partial y}:$$

Second derivative as matrix multiplication

```
Y = diff(X,2)
Y = [X(3)-2X(2)+X(1)...   X(m)-2X(m-1)+X(m-2)]
```

$$
\begin{bmatrix}
X(1)-2X(2) \\
X(1)-2X(2)+X(3) \\
X(2)-2X(3)+X(4) \\
. \\
. \\
. \\
. \\
X(m-2)-2X(m-1)+X(m) \\
X(m-1)-2X(m)
\end{bmatrix}
=
\begin{bmatrix}
-2 & 1 & 0 & 0 & & & & 0 \\
1 & -2 & 1 & 0 & & & & . \\
0 & 1 & -2 & 1 & 0 & & & . \\
0 & & 1 & -2 & 1 & & & \\
 & & & & -2 & & & \\
. & & . & & . & & & . \\
 & & & & & -2 & 1 & 0 \\
 & & & & & 1 & -2 & 1 \\
0 & & & & & .\ . & 1 & -2
\end{bmatrix}
\times
\begin{bmatrix}
X(1) \\
X(2) \\
X(3) \\
. \\
. \\
. \\
. \\
X(m-1) \\
X(m)
\end{bmatrix}
$$

Toeplitz matrix

Functional data

Coding second derivative using Toeplitz matrix

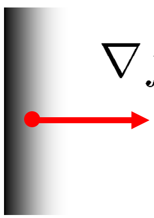
```
-2   1   0   0               0
 1  -2   1   0               .
 0   1  -2   1   0           .
 0       1  -2   1
                -2
 .       .       .           .
                    -2   1   0
                     1  -2   1
 0                  .  . 1  -2
```

```
c=zeros(1,T);
%first column
c(1:2)=[-2 1];
r=zeros(1,T);
%first row
r(1:2)=[-2 1];
```

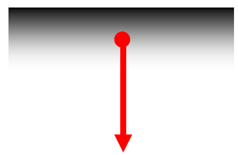
Image gradient

The *gradient* of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

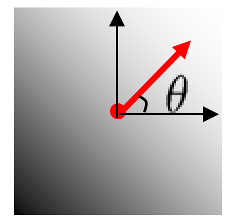
The gradient points in the direction of most rapid increase in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

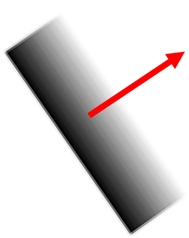


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient magnitude (edge strength): $$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

The gradient direction is given by: $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

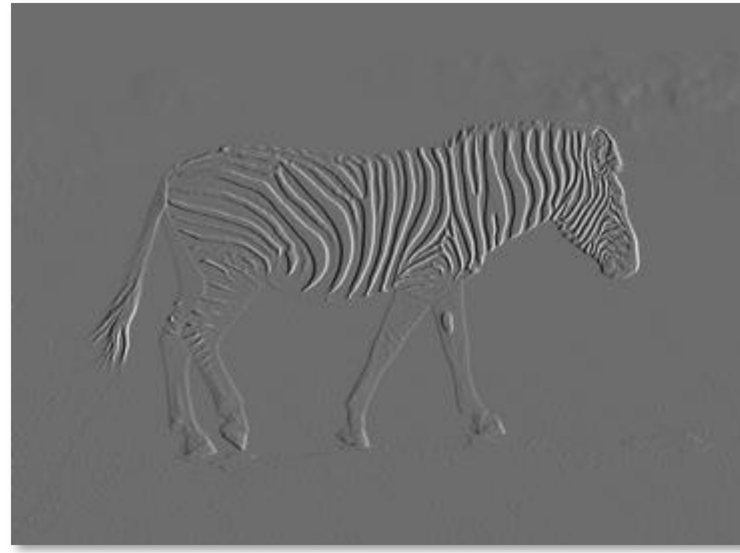
Exercise:



How to compute the gradient in this example?

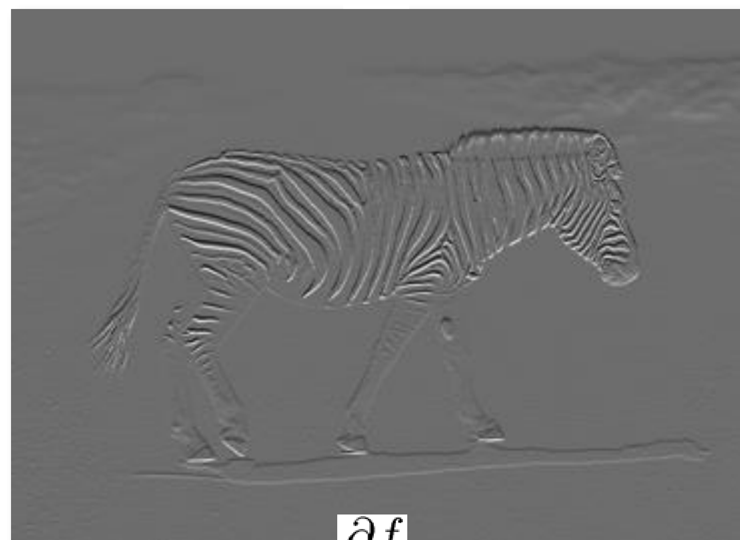
Image gradient

f



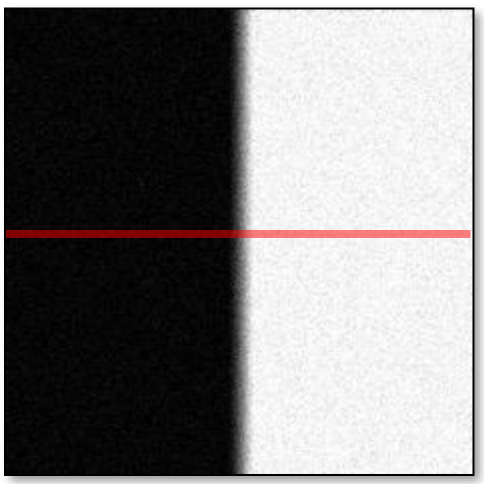
$$\frac{\partial f}{\partial x}$$

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



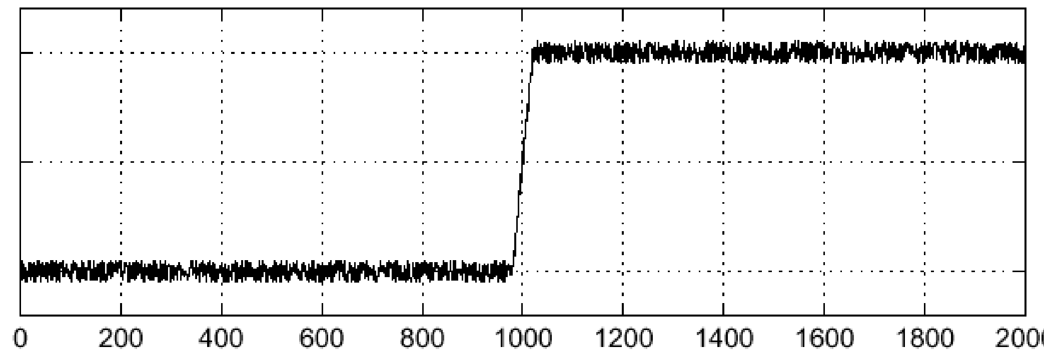
$$\frac{\partial f}{\partial y}$$

Robust estimation of derivatives

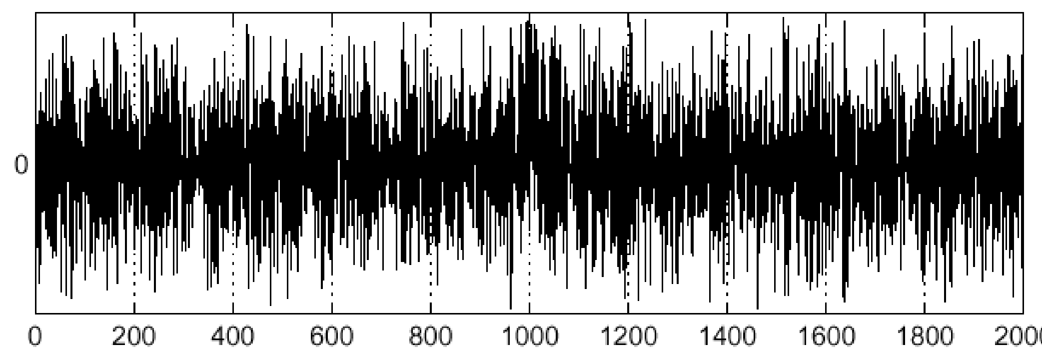


Noisy input image

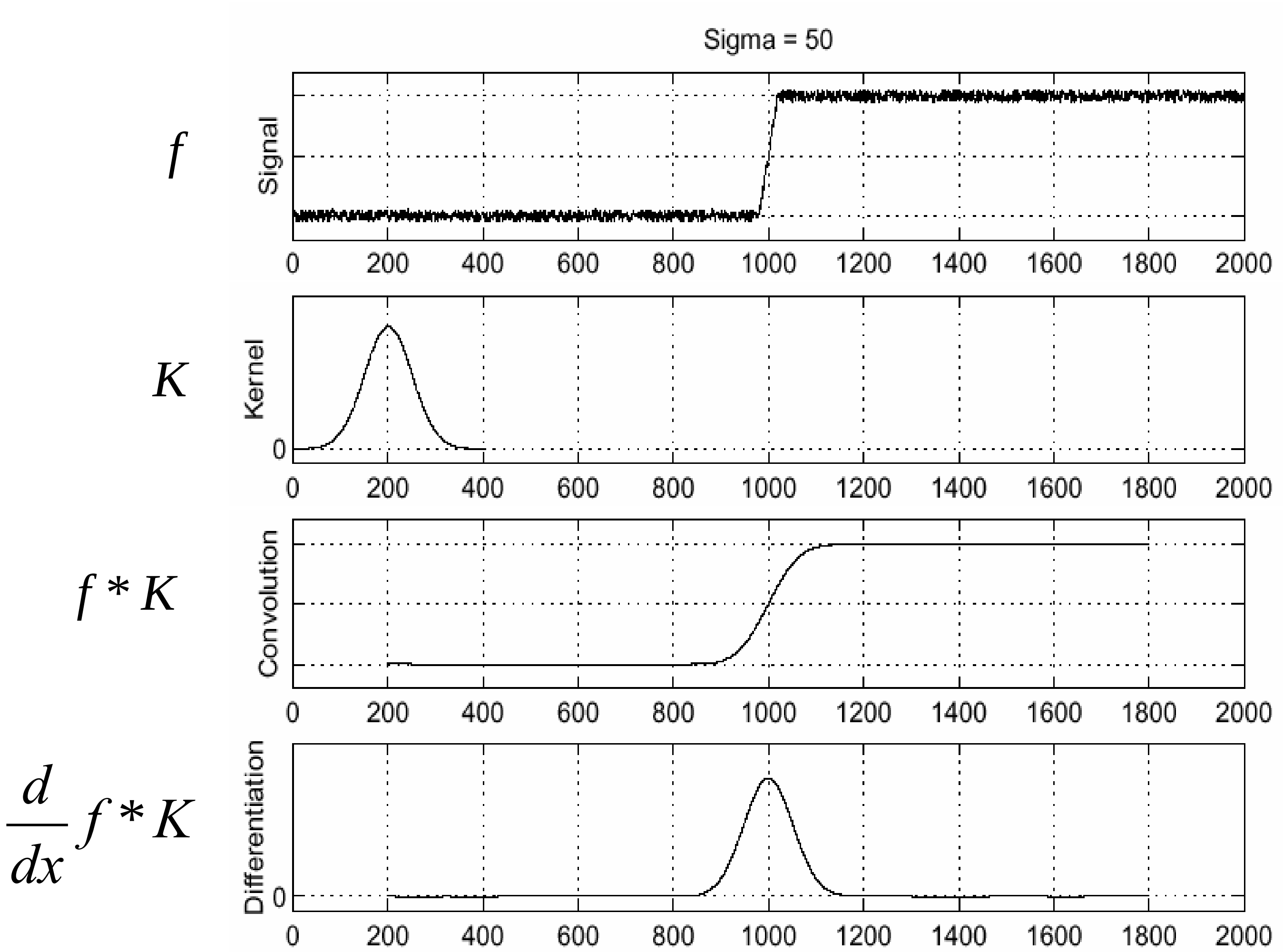
$f(x)$



$\frac{d}{dx}f(x)$



Kernel smoothing before derivative estimation



Question: How to stabilize gradient descent based optimization algorithms based on this figure?

Associative property of convolution

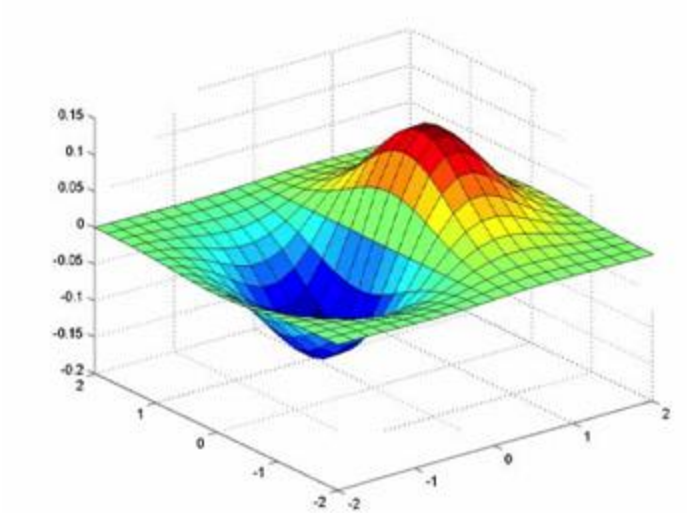
Differentiation is convolution, and convolution is associative:

$$\frac{d}{dx}(K*f)=\frac{d}{dx}\int K(x,y)f(y)\,dy=\frac{dK}{dx}*f$$

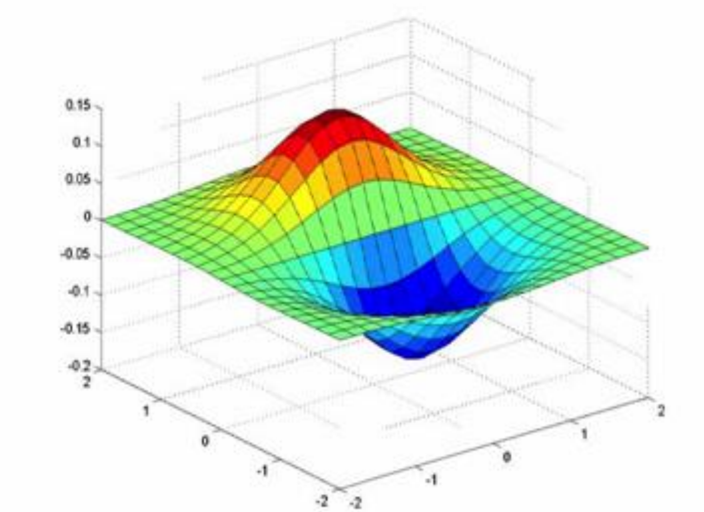
Why math matters in algorithm development?

This reduces one operation

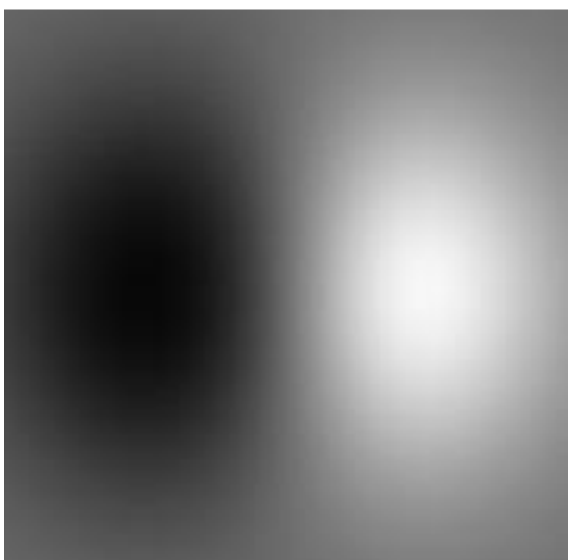
Derivative of Gaussian kernel

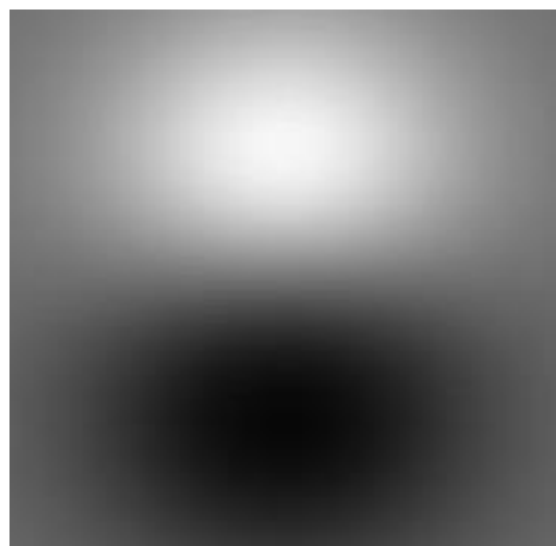


x-direction

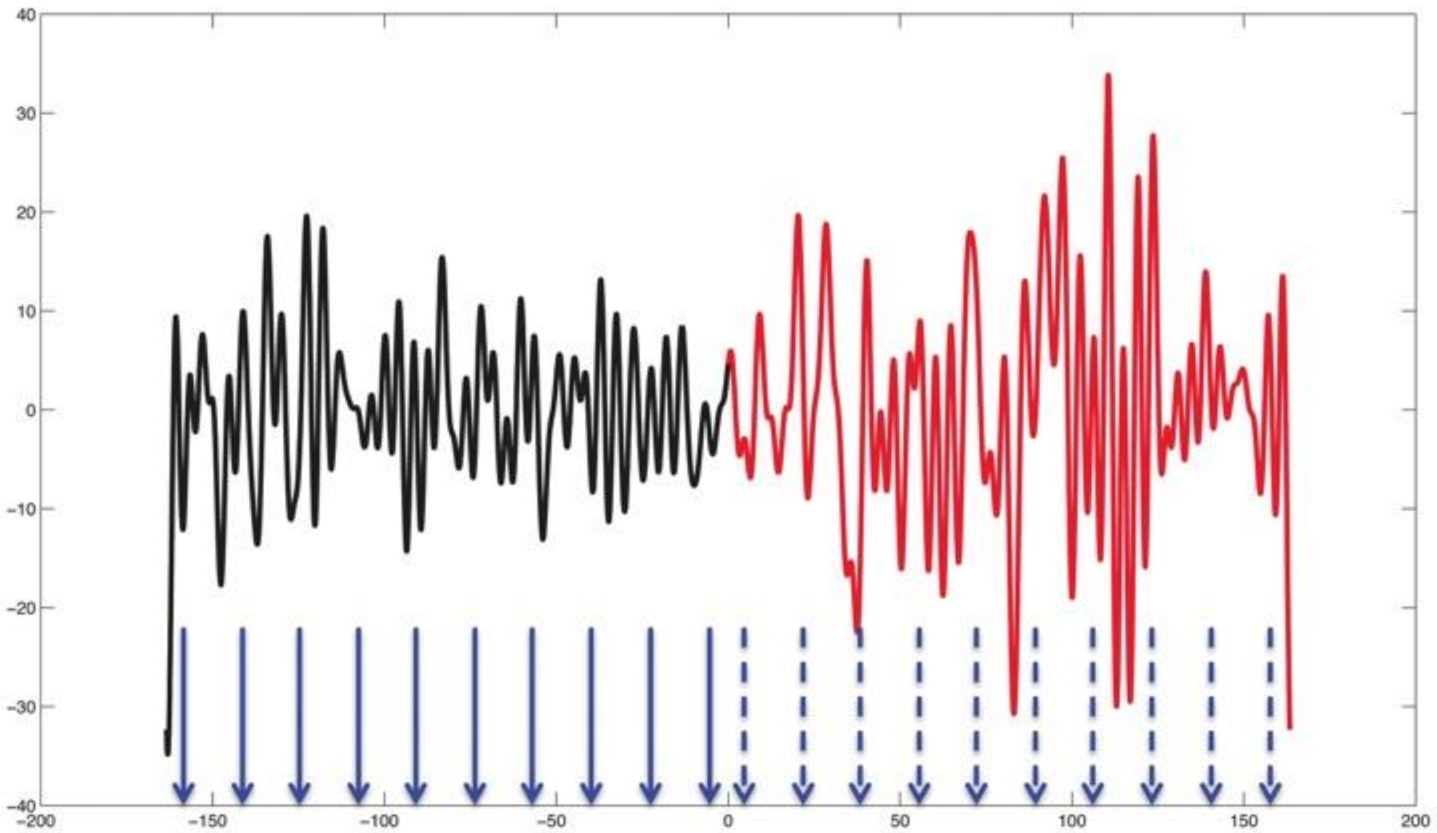


y-direction





Detect sudden changes in time series data



Black = pre-seizure Red = seizure attack

Wang et al. 2014 ENAR distinguished paper award, 2015 ISBI

www.stat.wisc.edu/~mchung/papers/wang.2015.ISBI.seizure.pdf

MATLAB Demo

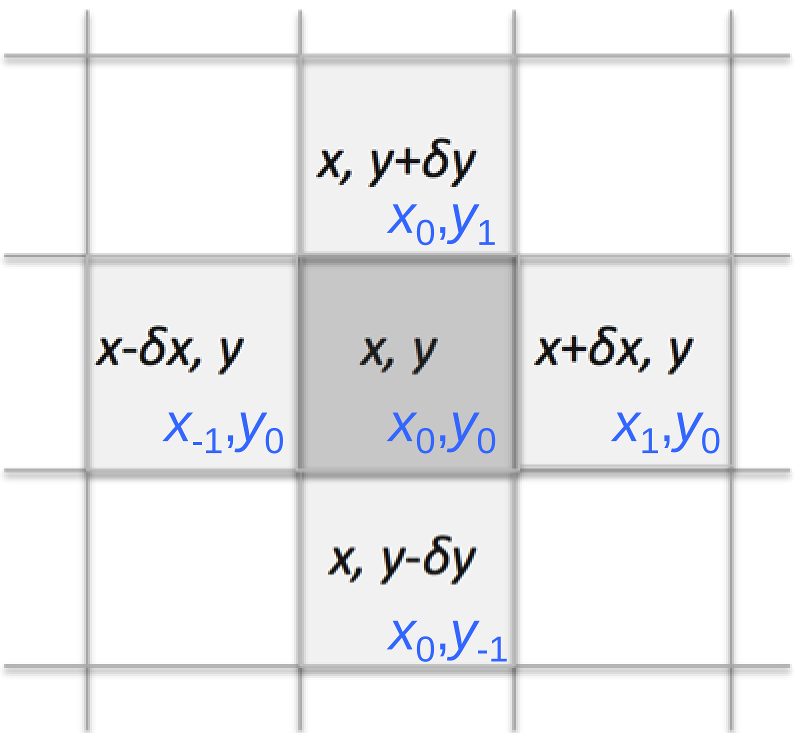
Differential Operators

Laplace Operator in 2D Euclidean space

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

When δx=δy=1,

$$\Delta f(x,y) = f(x+\delta x, y) + f(x, y+\delta y) + f(x-\delta x, y) + f(x, y-\delta y) - 4f(x,y)$$



$$\Delta f = \sum_{i,j=-1,0,1} w_{ij} f(x_i, y_i)$$

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

Exercise: Implement Laplacian as matrix multiplication

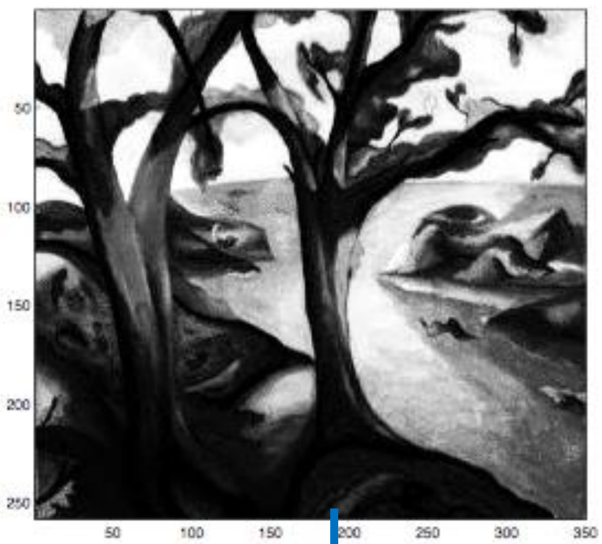
Application of Laplacian operator

Solving diffusion equations → denoising, regression

Obtaining orthonormal basis → spectral geometry

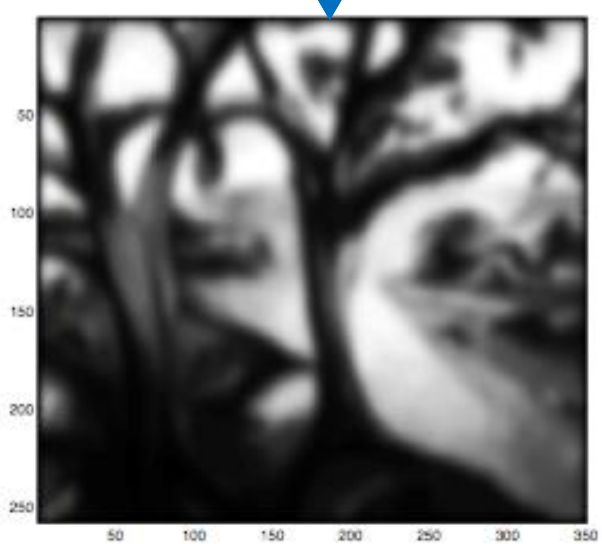
Direct application to images → boundary detection

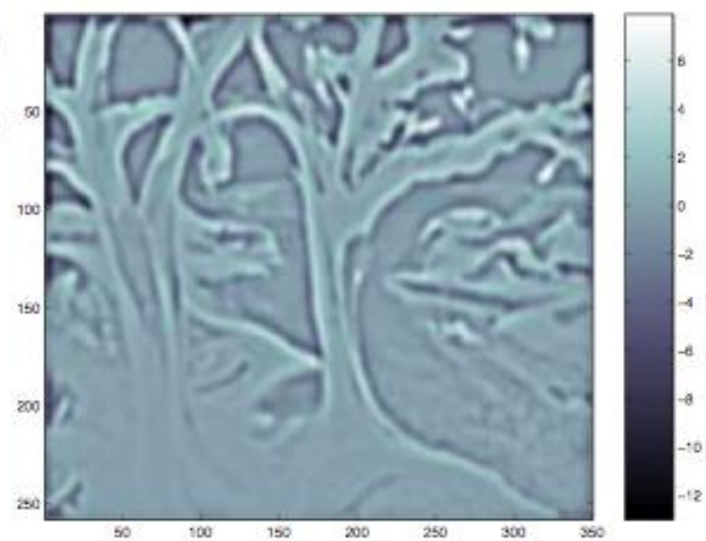
Edge detection using Laplacian



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

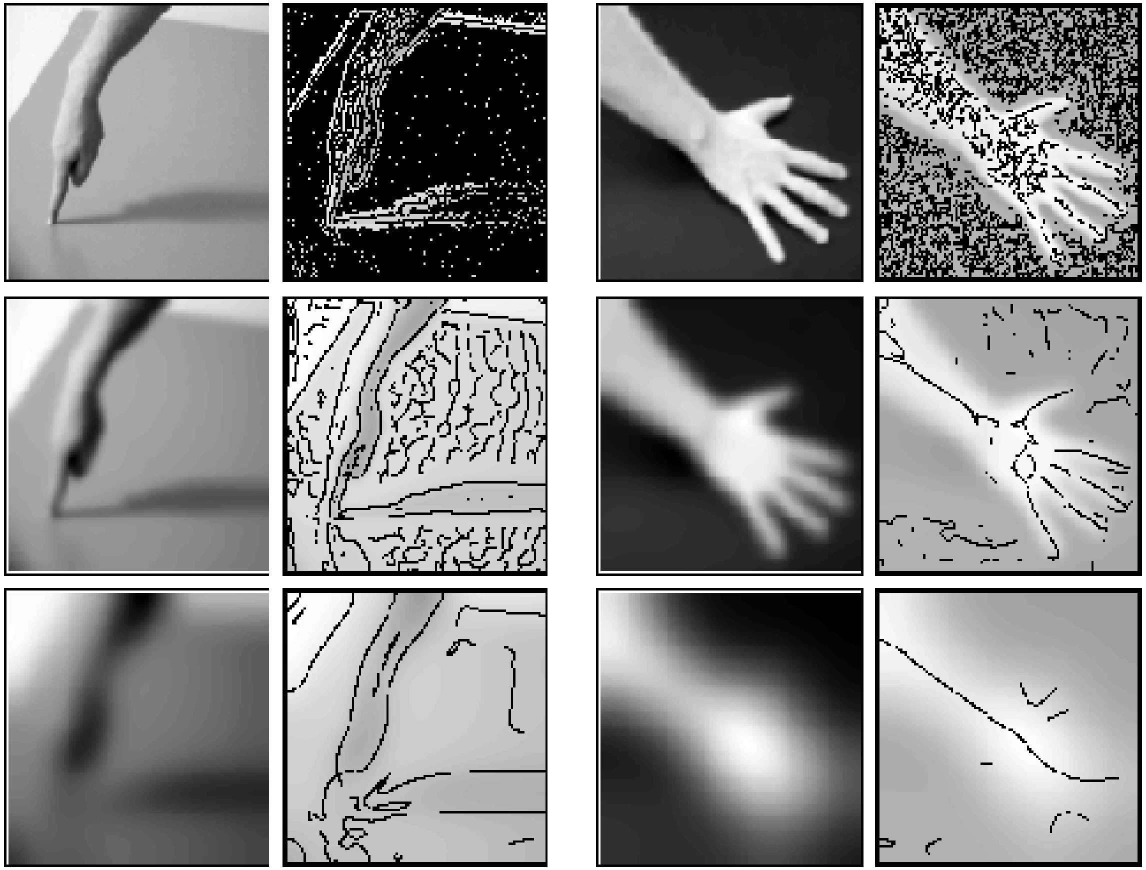
$$\Delta f = \sum_{i,j=-1,0,1} w_{ij} f(x_i, y_i)$$





Gaussian smoothing → Laplacian

Edge detection at different smoothing scales



Should we determine <u>optimal</u> smoothing scale?
Answer 50 years ago?
Answer now?

Edge detection using sample variance

Often simple Method performs far better than state-of-arts

Sample variance within moving window

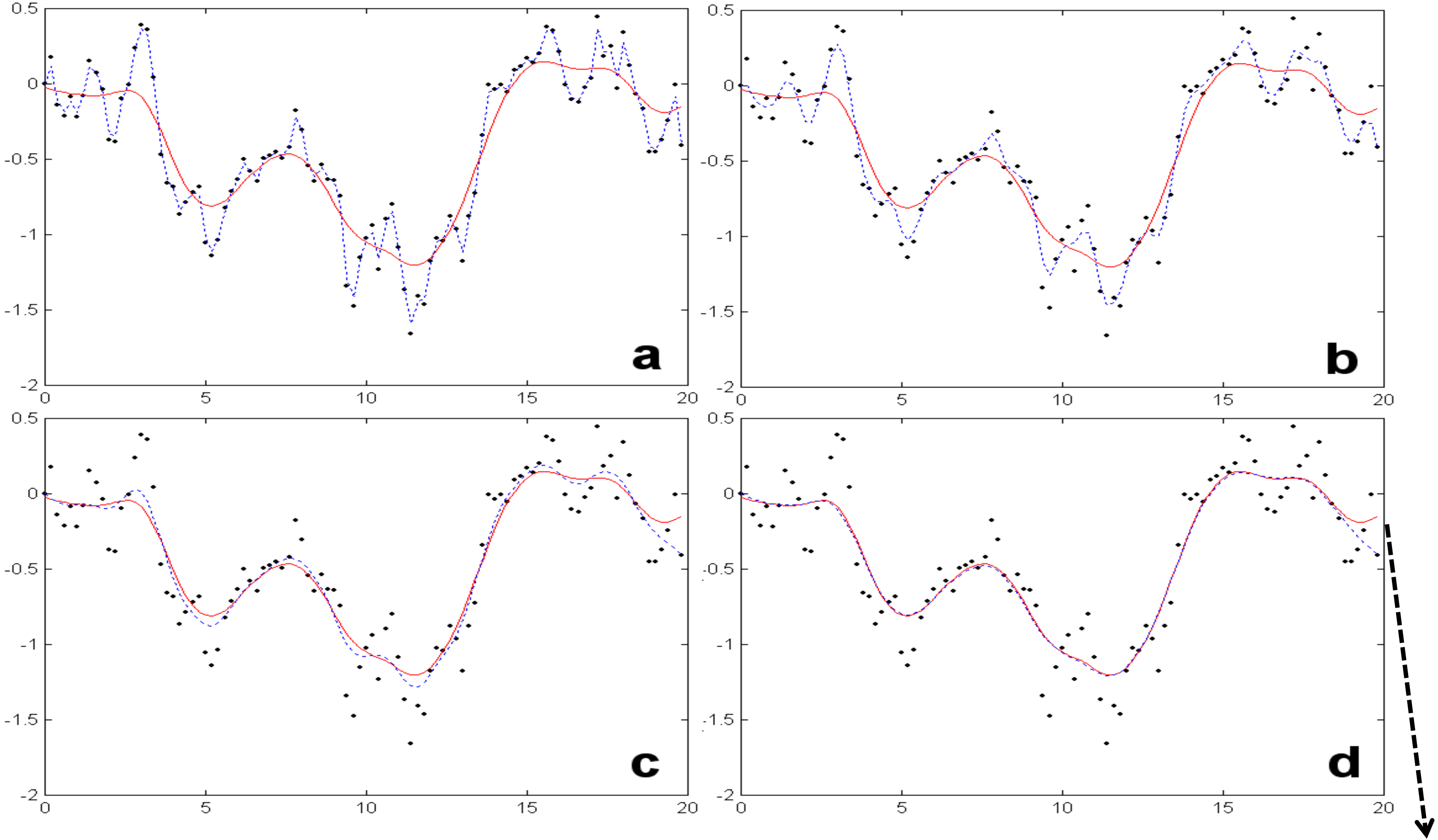
Without smoothing With smoothing

Diffusion Equation

Most important equation in image analysis

DE

Diffusion equation on 1D functional data



Shrinkage at the boundary

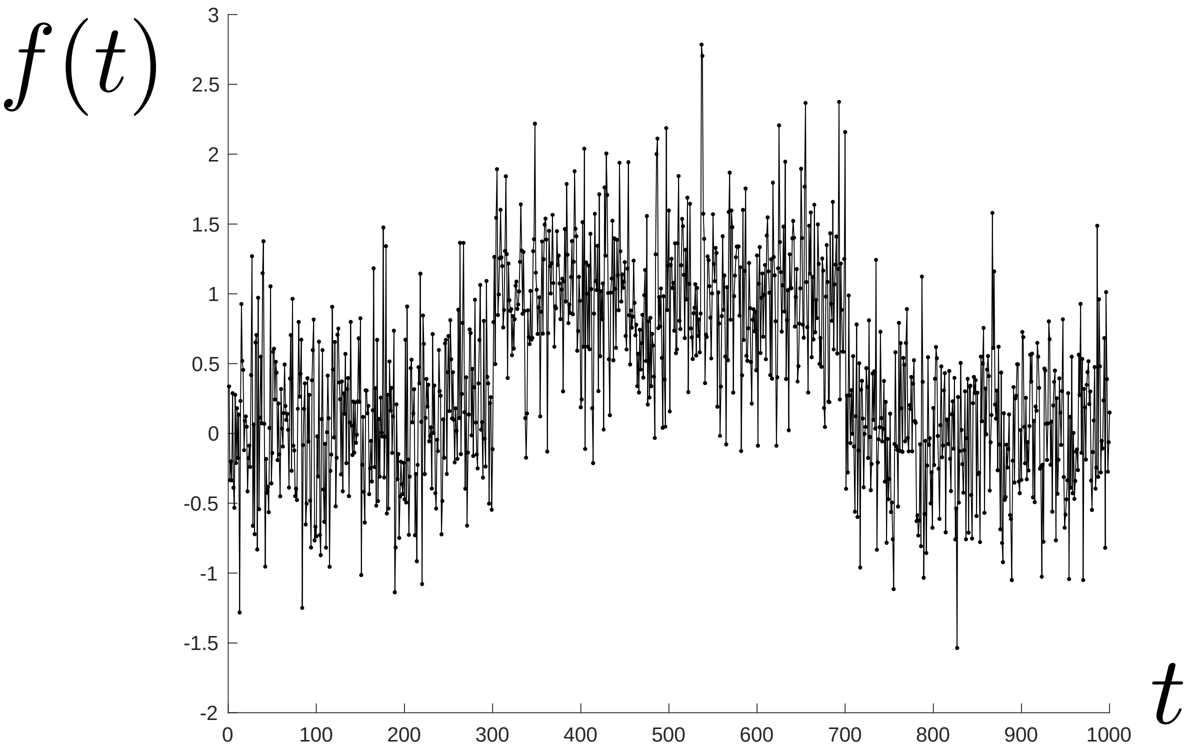
Red= Gaussian kernel convolution

Blue = Diffusion equation after 5, 25 and 50 iterations

Diffusion equation in Euclidean space

$$\frac{\partial g(\sigma, t)}{\partial \sigma} = \Delta g(\sigma, t) \qquad g(\sigma = 0, t) = f(t)$$

Initial condition



Let's solve it numerically.

mathematical (regression) approach will be studied later in GDA

Diffusion equation in Euclidean space

$$\frac{\partial g(\sigma,t)}{\partial \sigma} = \Delta g(\sigma,t) \qquad g(\sigma = 0, t) = f(t)$$

Initial condition

Iteration over dummy parameter sigma:

$$g(\sigma_{n+1},t) - g(\sigma_n,t) = (\sigma_{n+1} - \sigma_n)\Delta g(\sigma_n,t)$$

Discrete estimation

→ Matrix multiplication

Initial condition $\quad g(\sigma_0 = 0, t) = f(t)$

Iterative matrix multiplication algorithm

1) Discretize Laplacian as a matrix L

2) *Diffusion time* $s = 0$

3) Initial vector $g \leftarrow f$

4) $g \leftarrow g + ds * Lg$

5) *Diffusion time* $s \leftarrow s + ds$

Iterative matrix multiplication algorithm

```
L = [1 -2 1]                                   %Lapl
g=y;
for i=1:100
    Lg = conv(g, L, 'same');
    g = g+ 0.01*Lg;                            %total dif
end
```

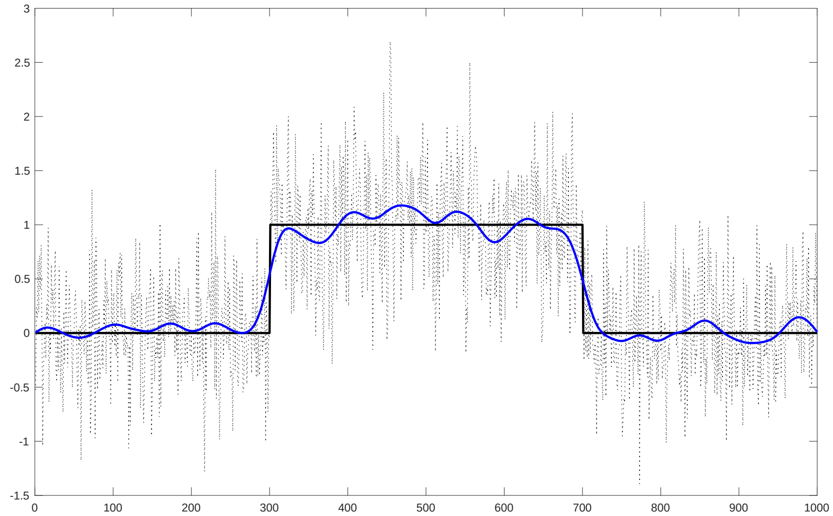
`L2` ← Toeplitz matrix

```
L2(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
```

```
g=y;
for i=1:1000
    Lg = L2*g;
    g = g+ 0.01*Lg;
end
```

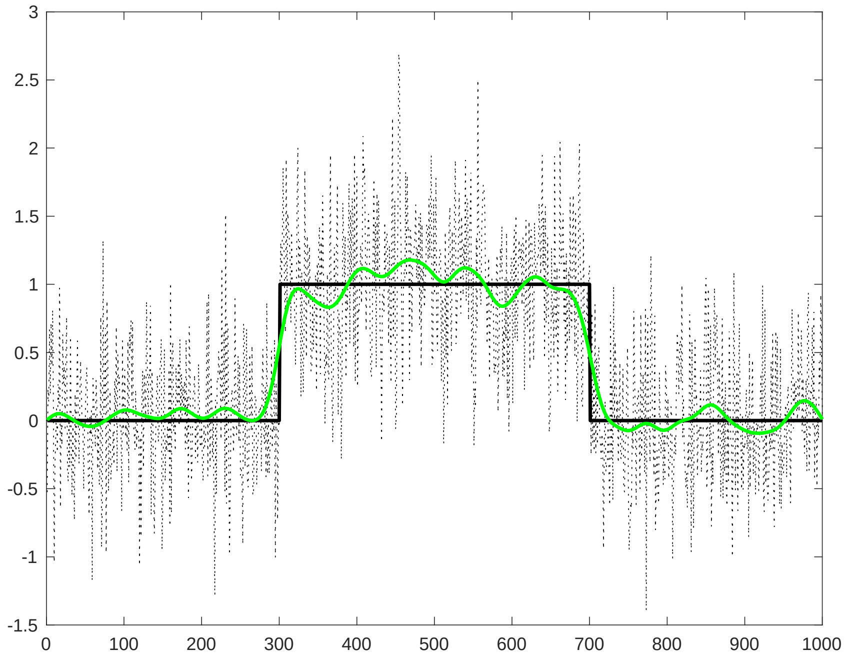
Matlab demo

Iterative matrix multiplication algorithm



`conv.m` result

they exactly match



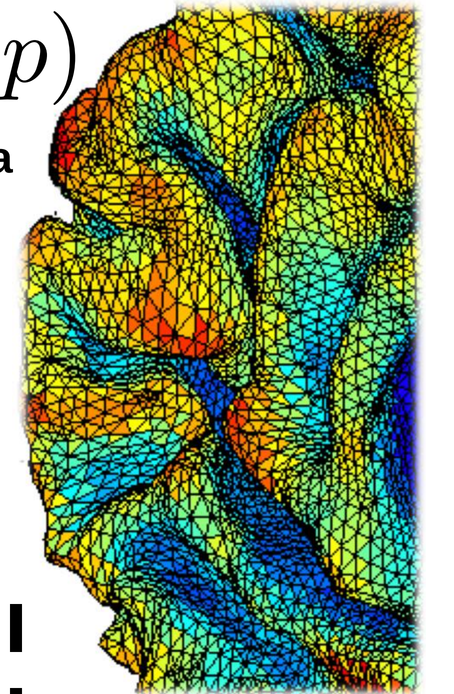
`toeplitz.m` result

Matlab demo

Diffusion on manifold

$f(p)$

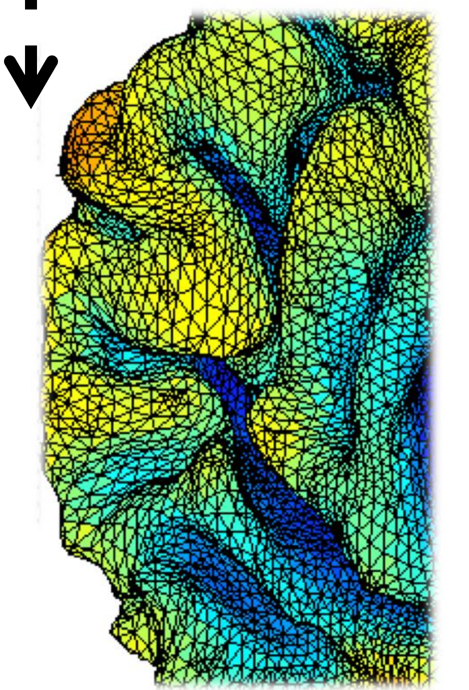
Initial data



$$\frac{\partial g(p,\sigma)}{\partial \sigma} + \Delta g = 0$$

$$g(p, \sigma = 0) = f(p)$$

Finite Element Method (FEM)
Finite Difference Method (FDM)



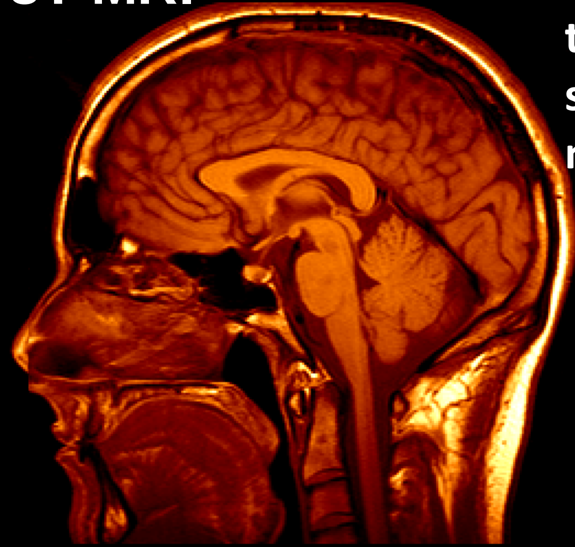
$g(p, \sigma)$

Smoothing

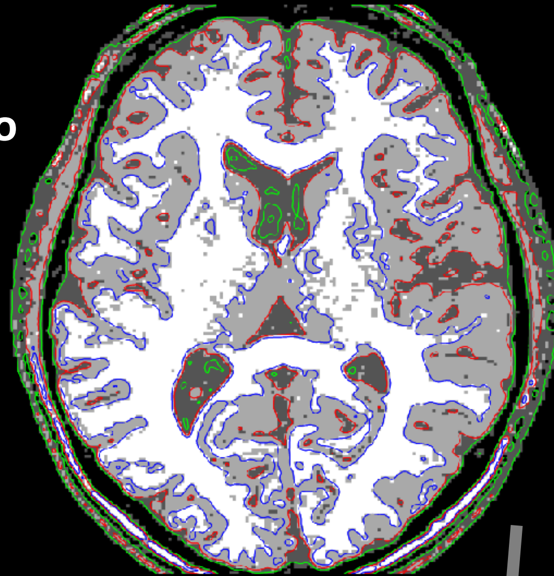
Chung et al. 2001 NeuroImage 13S:96

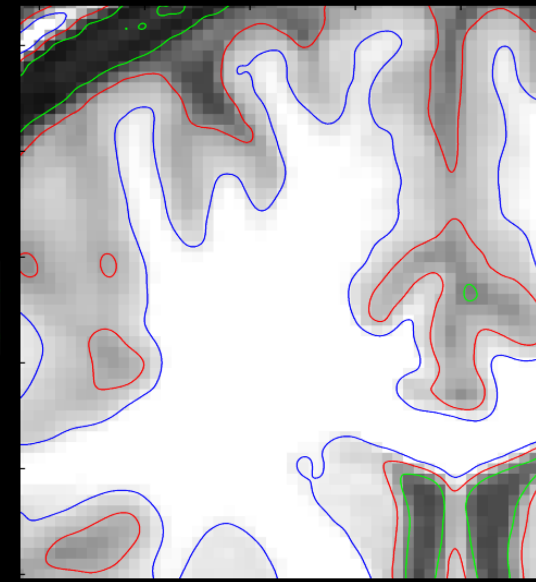
Laplace-Beltrami Operator

3T MRI

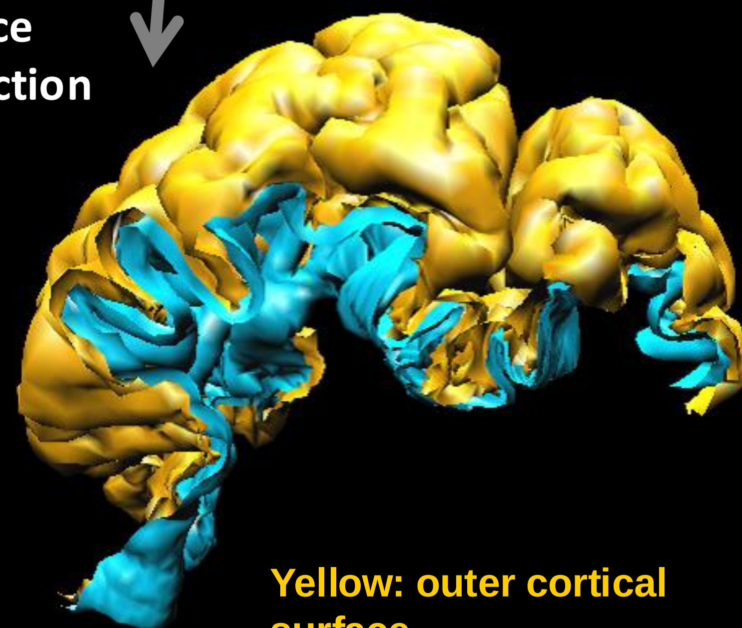


tissue
segmentatio
n





surface
extraction

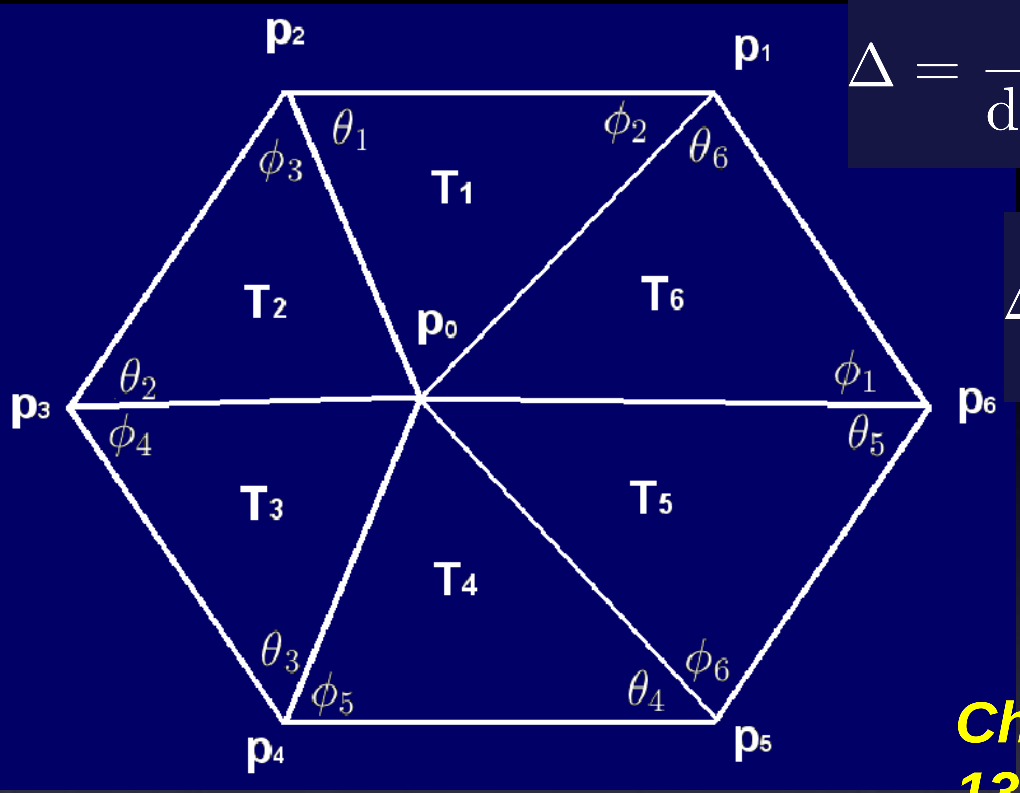


Yellow: outer cortical
surface
Blue: inner cortical
surface

MNI pipeline, McGill

Triangle mesh with 0.6
million triangles

Cotan formula for LB-operator on manifolds



$$\Delta = \frac{1}{\det g^{1/2}} \sum_{i,j} \frac{\partial}{\partial u^i} \left(\det g^{1/2} g^{ij} \frac{\partial}{\partial u^j} \right)$$

$$\Delta f(p) = \sum_{i=1}^{m} w_i [f(p_i) - f(p_0)]$$

$$w_j = \frac{\cot \theta_j + \cot \phi_j}{\sum_{j=1}^{m} |T_i|}$$

Chung et al. 2001 NeuroImage 13S:96

The first cotan formula

Proof by differential forms:

Lopez-Perez et al, 2004, ECCV

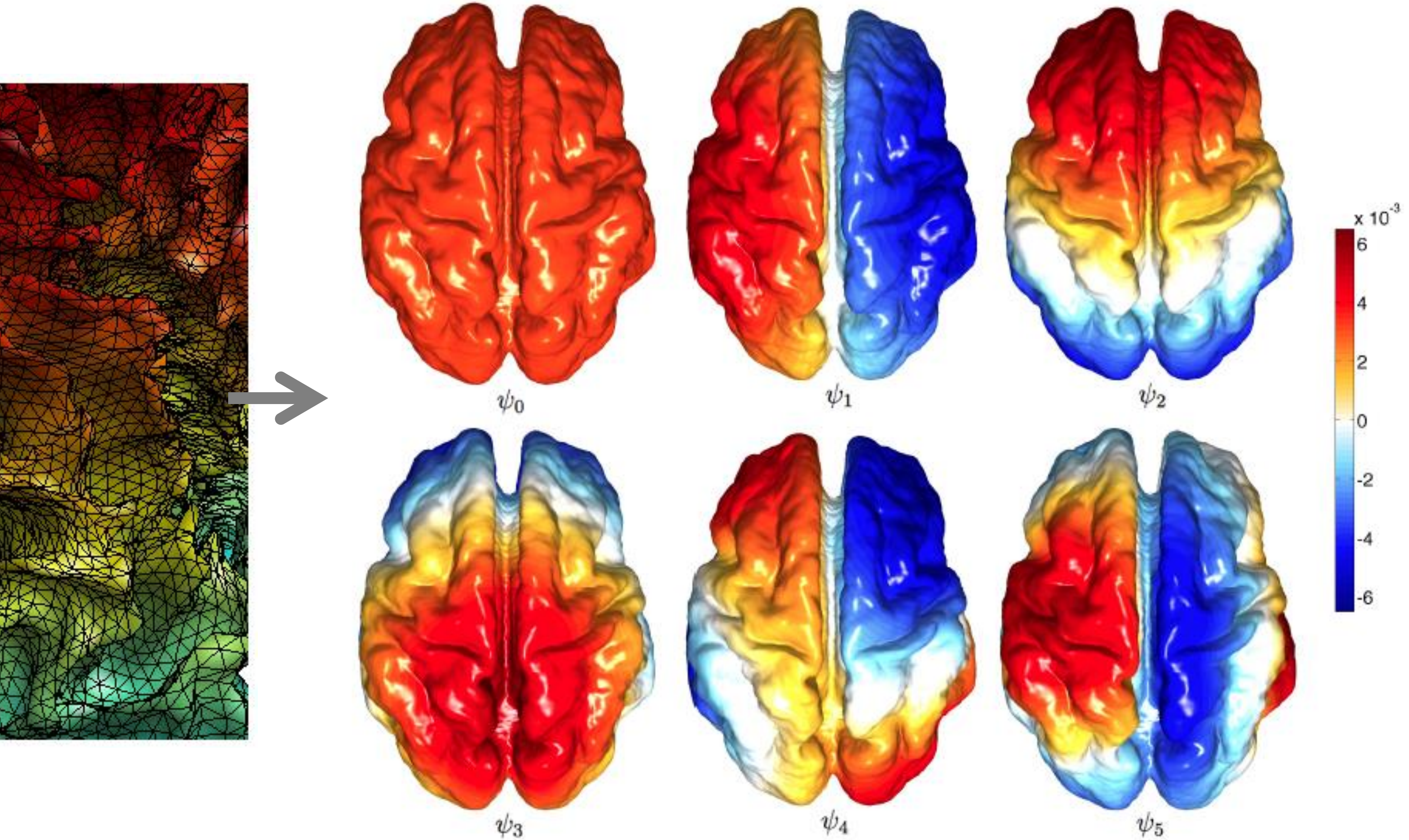
Chung et al. 2003 CVPR 467-473

Code

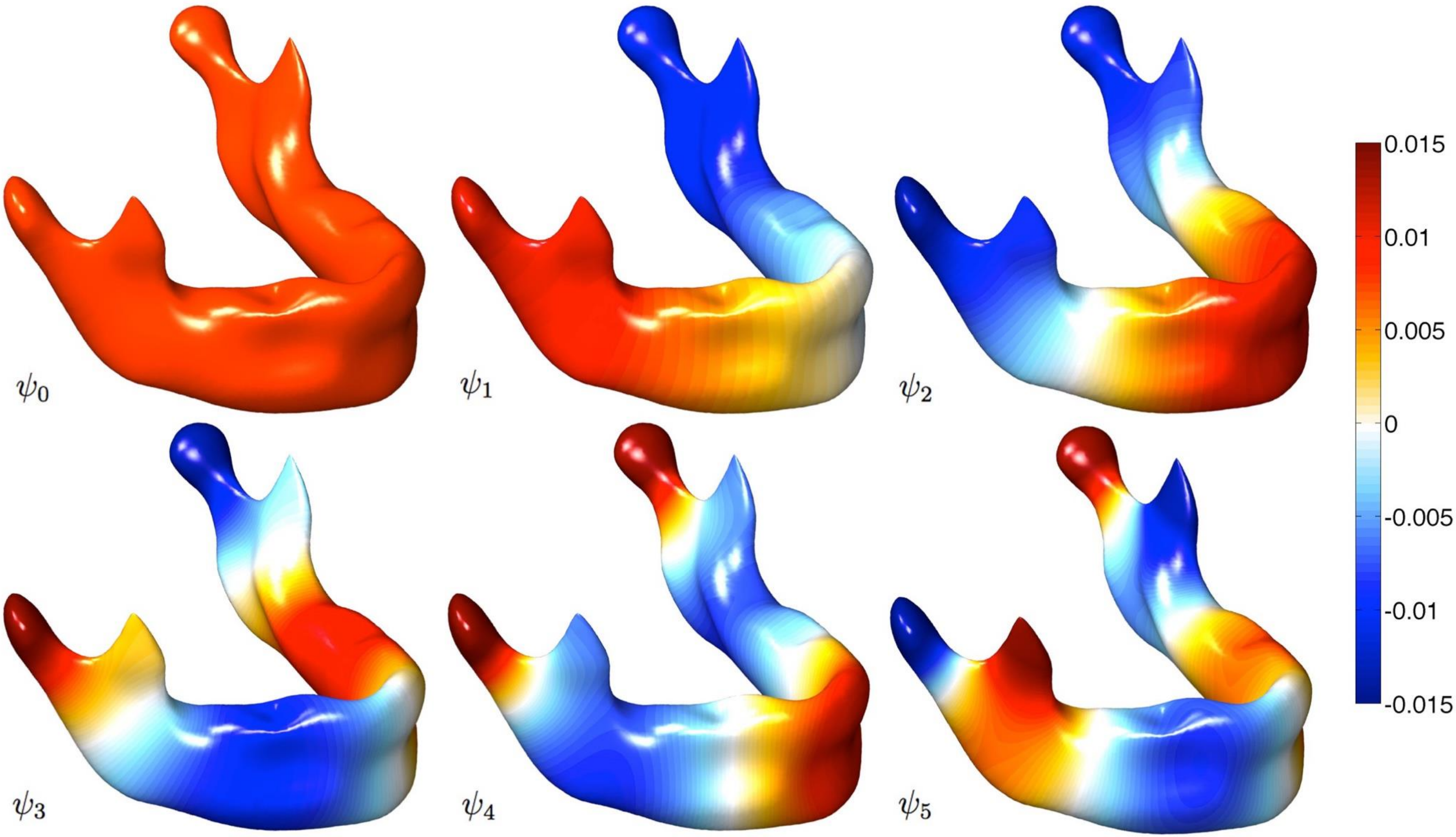
http://brainimaging.waisman.wisc.edu/~chung/lb

Chung & Taylor 2004 ISBI 432-435 FEM derivations

LB-eigenfunctions on brain surface $\Delta f = \lambda f$

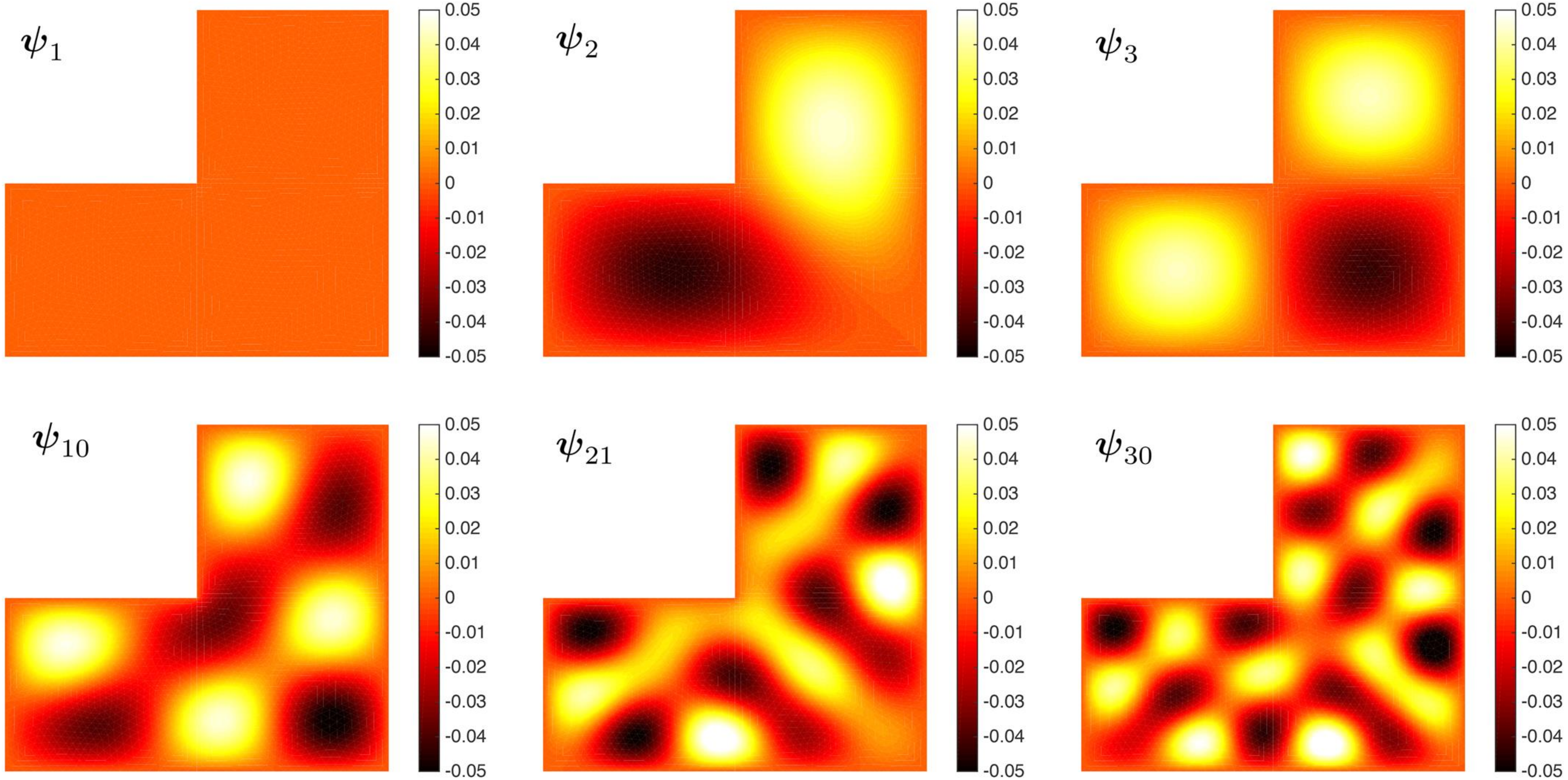


LB-eigenfunctions on mandible $\Delta f = \lambda f$

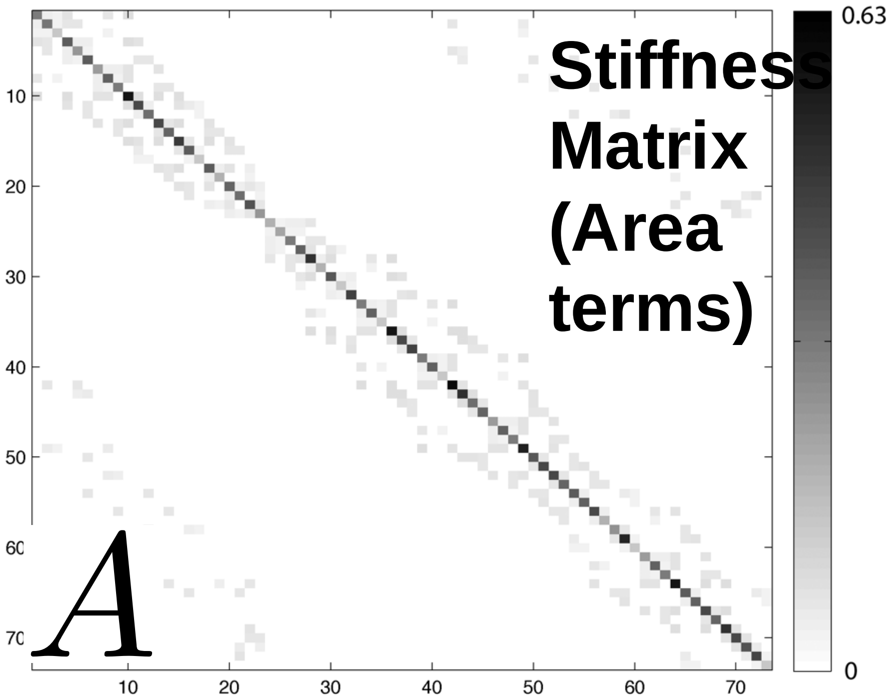


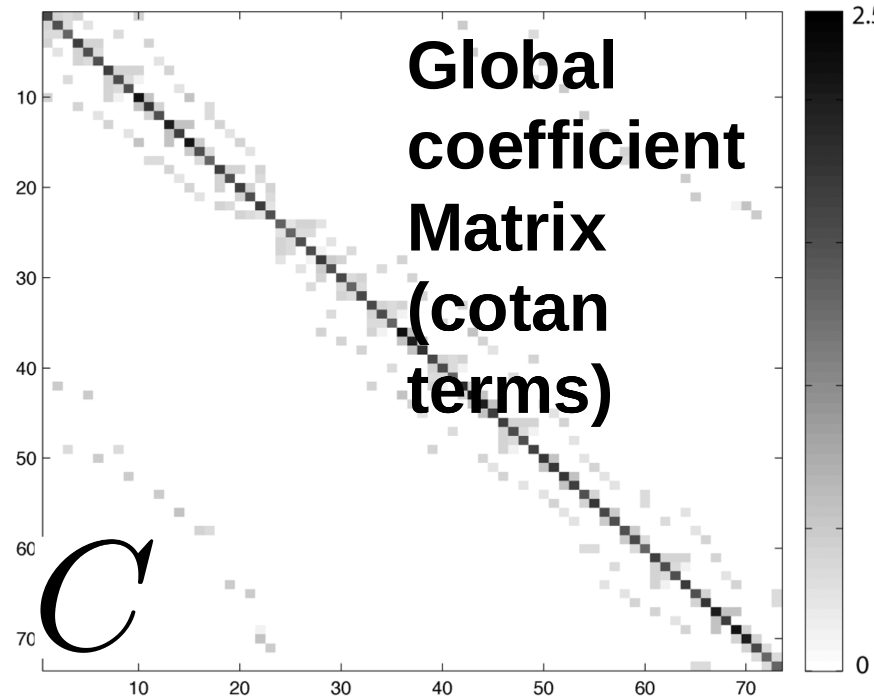
Seo et al. 2010 MICCAI 6363:505-512
Chung et al. 2015 Medical Image Analysis 22:63-76

LB-eigenfunctions in irregular domains



Discretization of diffusion equation → $\frac{d\mathbf{f}}{dt} = -A^{-1}C\mathbf{f}$



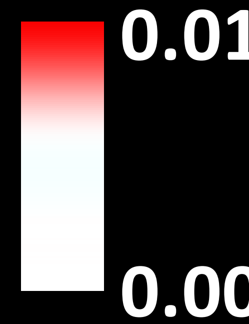


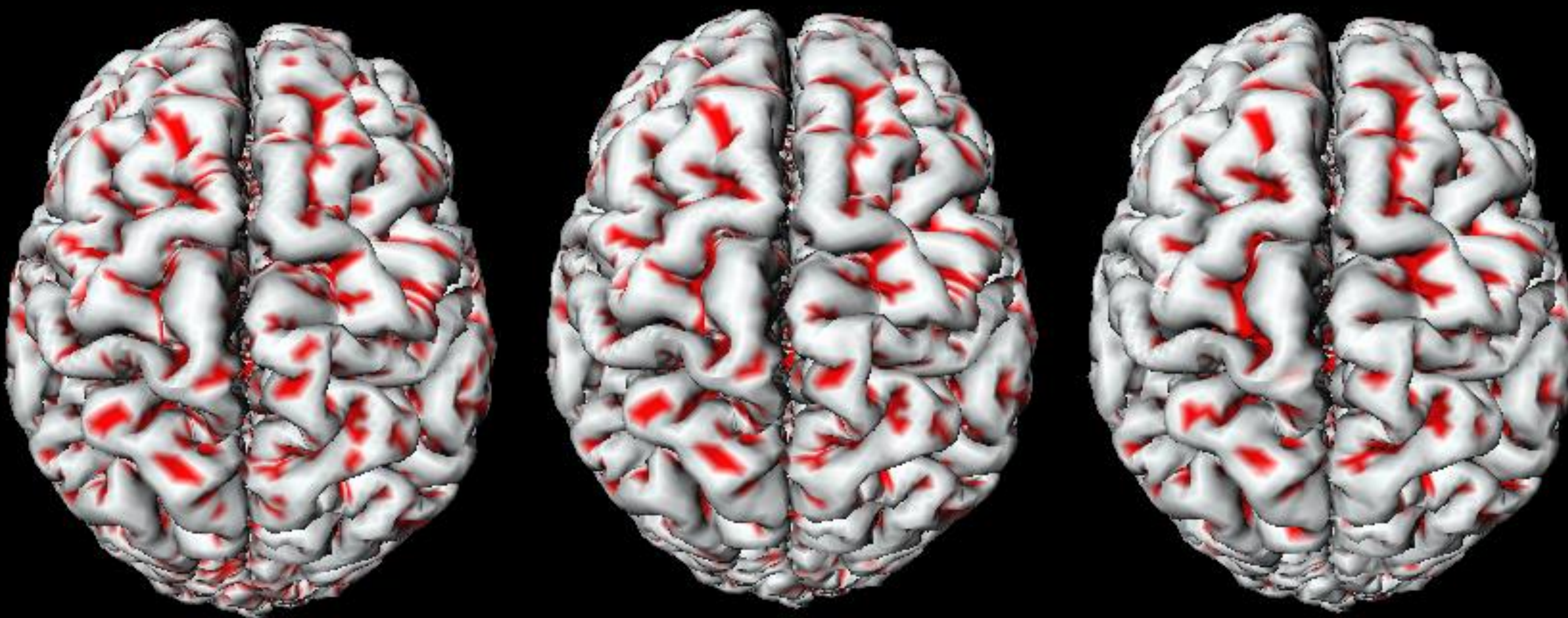
Tested up to two million mesh vertices

Chung & Taylor 2004 ISBI 432-435

Code:

Diffusion done by finite difference

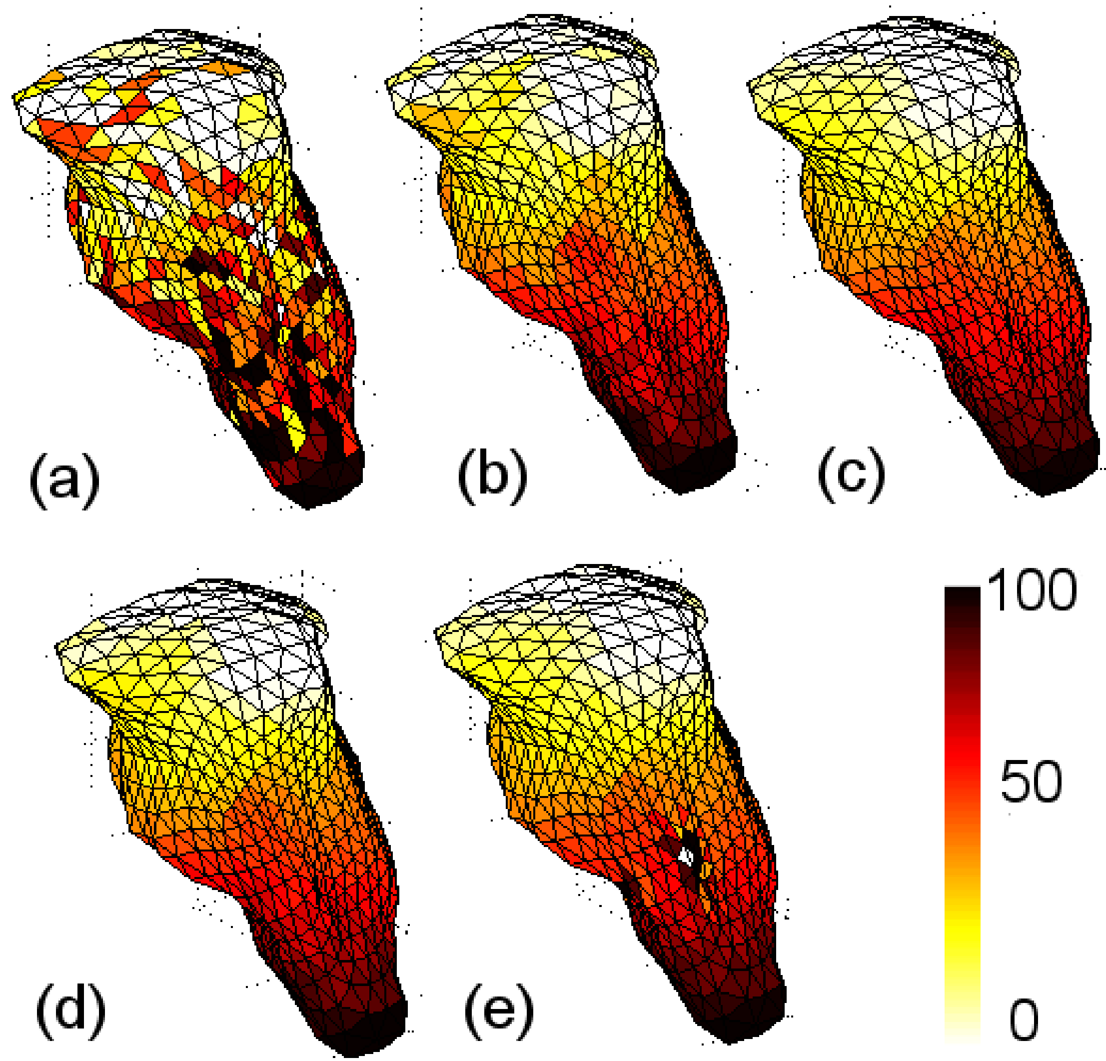




mean curvature 20 iterations 100 iterations

Code:

Finite Difference Scheme (forward Euler scheme) sensitive to step size



Chung et al. 2003 CVPR 467-473

Chung et al. 2003 NeuroImage 18:198-

Heat kernel smoothing

$$\frac{\partial g(p,\sigma)}{\partial \sigma} + \Delta g = \delta(p)$$

Dirac delta

Fundamental solution $g(p,\sigma) = K_\sigma(p,q)$

Probability density

$$\int_{\mathcal{M}} K_\sigma(p,q)\, d\mu(p) = 1$$

Initial condition

$$g(p,\sigma=0) = f(p)$$

General solution $g(p,\sigma) = K_\sigma * f(p) = \int_{\mathcal{M}} K_\sigma(p,q) f(p)\, d\mu(q)$

Chung et al. 2005 IPMI 3565:627-638
Chung et al. 2005 NeuroImage 25:1256-1265

First paper introducing heat kernel smoothing to

kernel

$$\Delta\psi_j = \lambda_j\psi_j \dashrightarrow \langle\psi_i,\psi_j\rangle = \delta_{ij}$$

$$K_\sigma(p,q) = \sum_{j=0}^{\infty} e^{-\lambda_j\sigma}\psi_j(p)\psi_j(q)$$

$$g(p,\sigma) = K_\sigma * f(p) = \int_{\mathcal{M}} K_\sigma(p,q) f(p)\, d\mu(q)$$

$$= \sum_{j=0}^{\infty} e^{-\lambda_j\sigma} f_j\psi_j(p)$$

Fourier coefficients

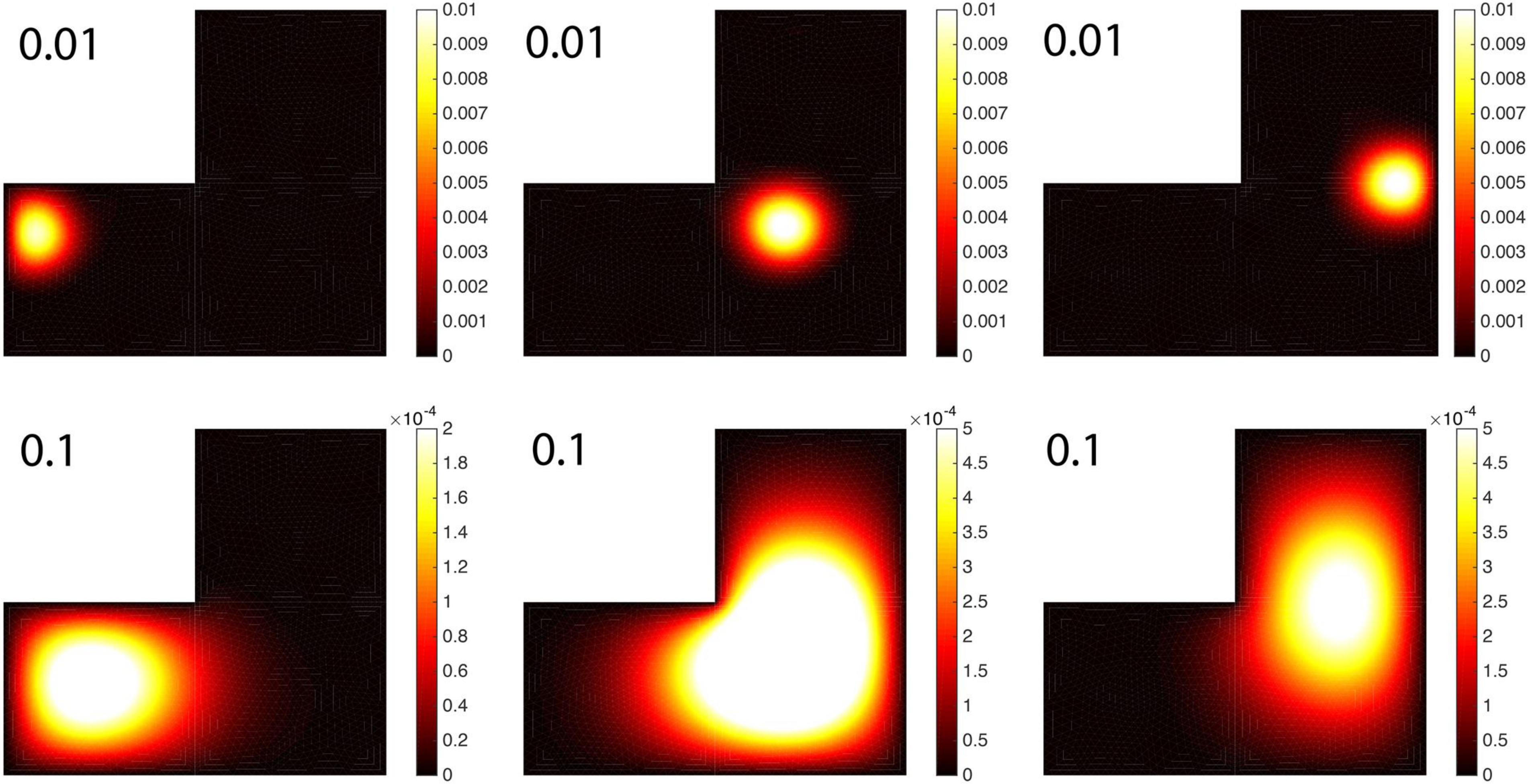
$$f_j = \int_{\mathcal{M}} f(p)\psi_j(p)\, d\mu(p)$$

Seo et al. 2010 MICCAI **6363:505-512**

Chung et al. 2015 Medical Image Analysis 22:63-76

Heat kernel

$$K_\sigma(p,q) = \sum_{j=0}^{\infty} e^{-\lambda_j \sigma} \psi_j(p)\psi_j(q)$$



Chung et al. 2005 IPMI 3565:627-638

Chung et al. 2005 NeuroImage 25

Polynomial approximation

IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 39, NO. 6, JUNE 2020 2201

Fast Polynomial Approximation of Heat Kernel Convolution on Manifolds and Its Application to Brain Sulcal and Gyral Graph Pattern Analysis

Shih-Gu Huang, Ilwoo Lyu, Anqi Qiu, and Moo K. Chung

Huang. 2020 IEEE TMI 39:2201-2212

Orthogonal polynomial $P_n(\lambda)$

$$\int_a^b P_n(\lambda) P_k(\lambda)\, d\lambda = \delta_{nk}$$

Second order recurrence

$$P_{n+1}(\lambda) = (A_n\lambda + B_n)P_n(\lambda) + C_n P_{n-1}(\lambda)$$

Jacobi polynomials (most general polynomial basis): Chebyshev, Hermite, Laguerre

Polynomial expansion of Laplace-Beltrami o

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

$$c_{\sigma,n} = \int_0^{\infty} e^{-\lambda\sigma} P_n(\lambda)\, d\lambda$$

$$\boxed{= \frac{\sigma^n}{(\sigma+1)^{n+1}}}$$

$$\Delta\psi_j(p) = \lambda_j \psi_j(p)$$

Diffusion given by heat kernel smoothing

Laguerre polynomial

$$K_\sigma * f(p) = \sum_{j=0}^{\infty} e^{-\lambda_j \sigma} f_j \psi_j(p) = \sum_{n=0}^{\infty} c_{\sigma,n} \sum_{j=0}^{\infty} P_n(\lambda_j) f_j \psi_j$$

$$= \sum_{n=0}^{\infty} c_{\sigma,n} P_n(\Delta) f(p)$$

Recurrence

$$\boxed{(n+1)P_{n+1}(x) = (2n+1-x)P_n(x) - nP_{n-1}(x)}$$

Matlab: Polynomial expansion of Laplacian

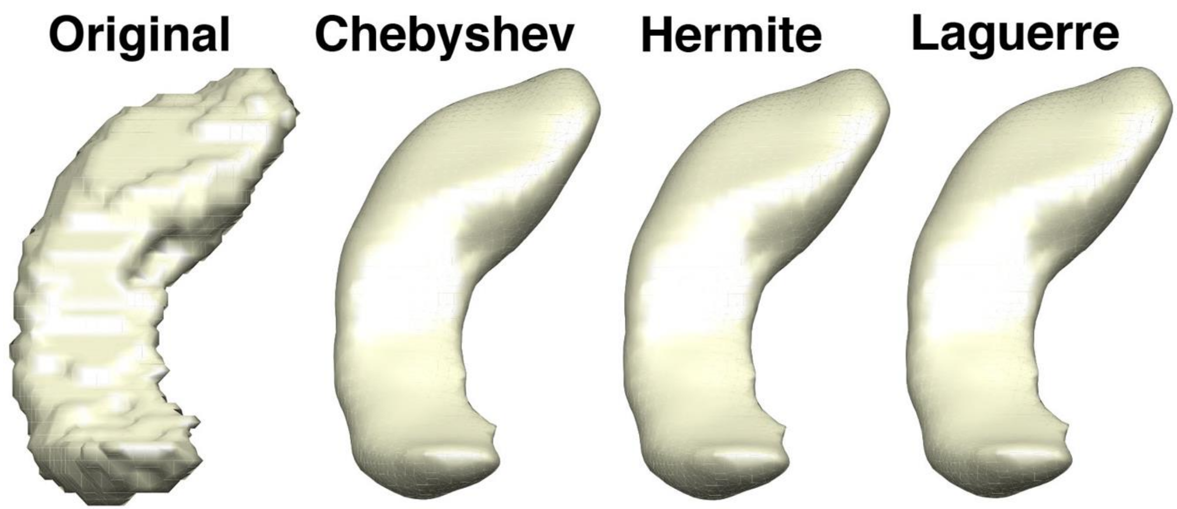
```
function g = heat_laguerre(f,L,t,m)

n=[0:m]';
coeff =(t/(t+1)).^n/(t+1);

Pf_old =0; Pf =f; g=0;

for n=0:m
     g =g +coeff(n+1)*Pf;
     Pf_new =(-L*Pf +Pf*(2*n+1) - Pf_old*n
)/(n+1);
     Pf_old =Pf;
     Pf =Pf_new;
End
```

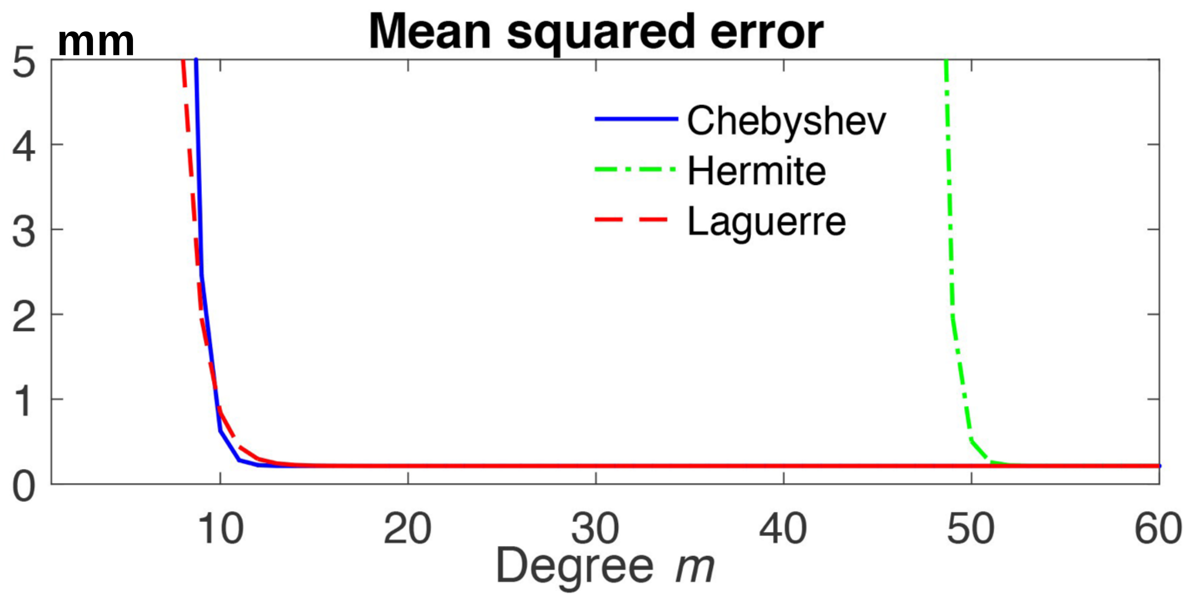
Run time

$\mathcal{O}(1)$

Huang et al. 2020, IEEE TMI 39:2201-2212

Polynomial approximation of surface coordinates (m=100, σ = 1.5)



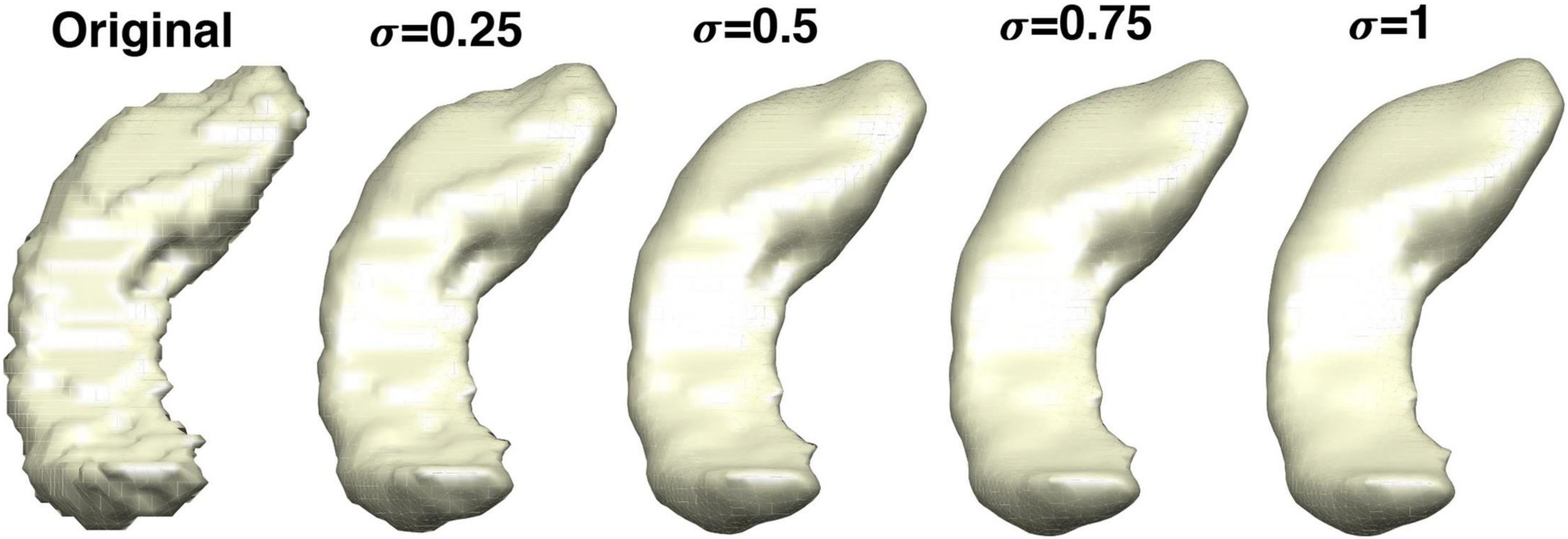


Huang et al. 2019 MICCAI 48-56

Code

Iterative heat kernel convolution on polynomial approximation

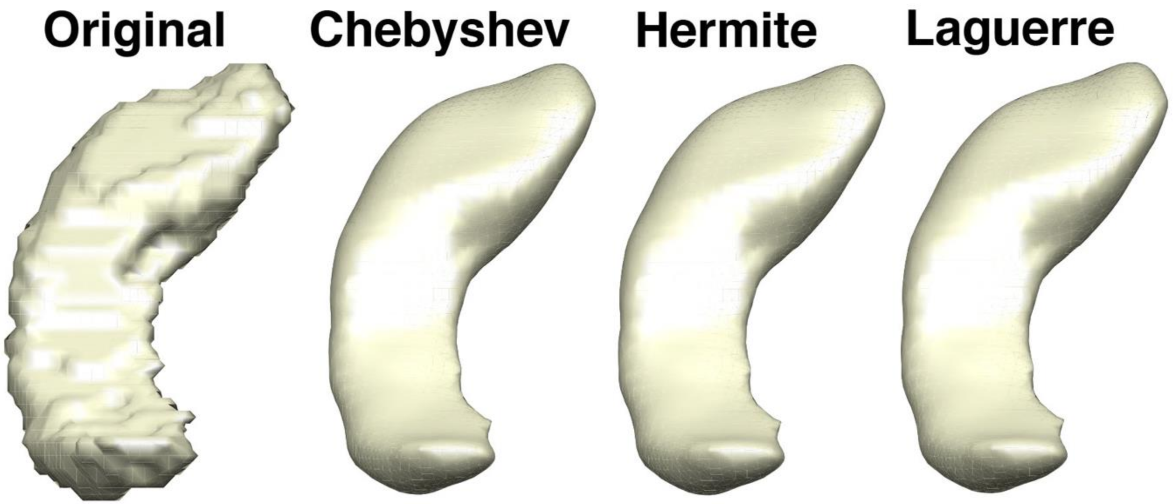
$$K_{\sigma_1+\sigma_2+\cdots+\sigma_m} * f = K_{\sigma_1} * K_{\sigma_2} * \cdots * K_{\sigma_m} * f$$

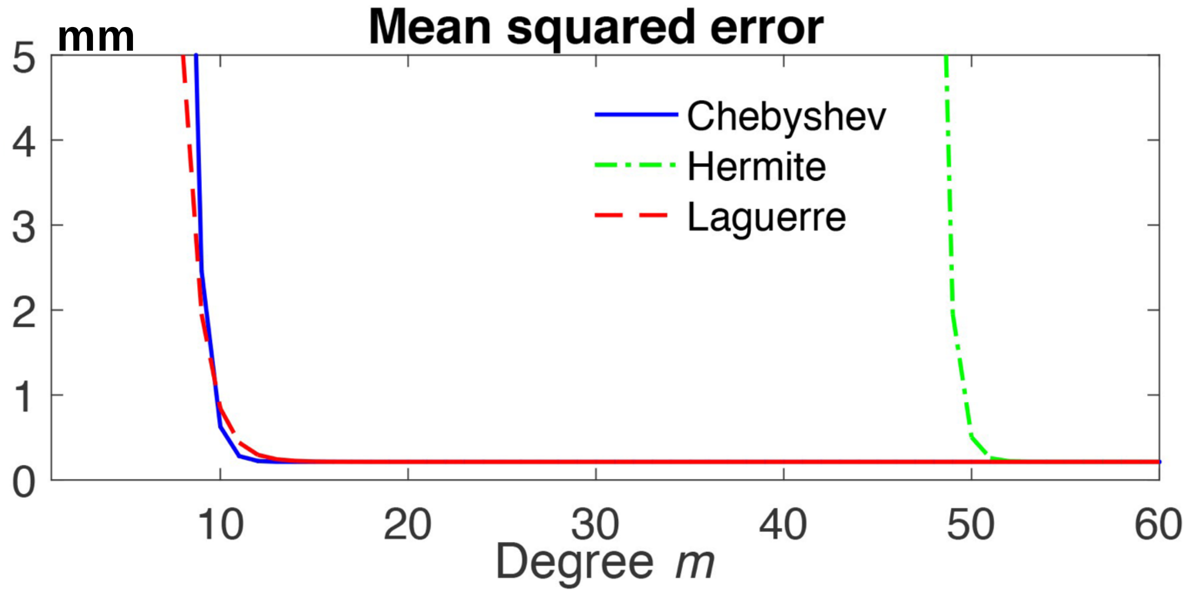


Huang et al. 2019 MICCAI 48-56

Code

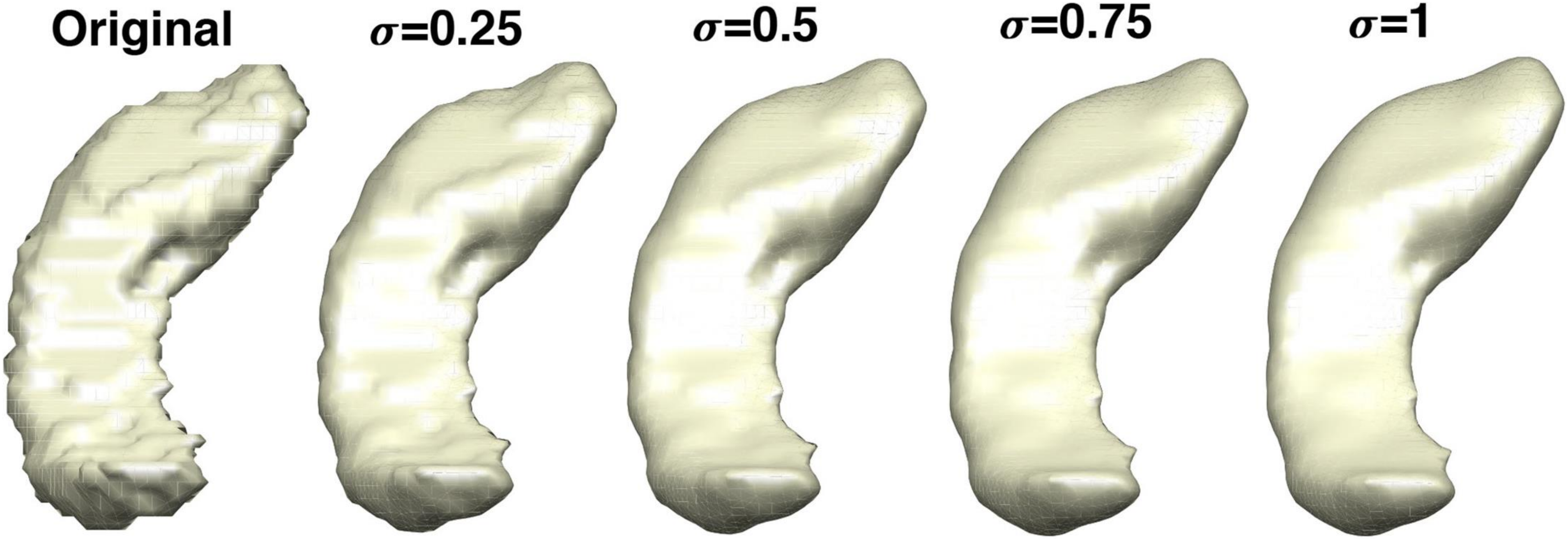
Polynomial approximation of surface coordinates (m=100, σ = 1.5)





Huang et al. 2019 MICCAI 48-56

Iterative heat kernel convolution on polynomial approximation



Huang et al. 2019 MICCAI 48-56