Stat 992: Lecture 26
Curve modeling II.

Moo K. Chung mchung@stat.wisc.edu

March 30, 2004

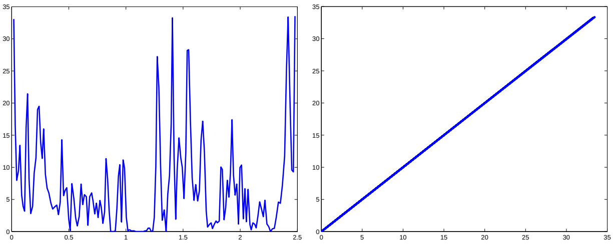

Figure 1: Left: curvature function $\kappa(\lambda)$ via Shubing's method. Right: finite difference method. Right: scatter plot of curvature estimations. Two methods gives almost identical estimation within 0.013% difference.

We lecture is based on **Problem 25**.

1. *Direct curvature estimation.* (Shubing Wang) The curvature is given by

$$\kappa = \frac{4A(\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1})}{|\mathbf{p}_{i-1} - \mathbf{p}_i||\mathbf{p}_i - \mathbf{p_{i+1}}||\mathbf{p}_{i+1} - \mathbf{p}_{i-1}|}$$

where $A(\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1})$ is the area of triangle with vertices $\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$.

```
%Modified from Shubing's code
load CCcurve.data
CC=[CCcurve(171,:);CCcurve;CCcurve(1,:)]
for i=2:172
 kappa2(i-1)=4*polyarea(CC((i-1):(i+1)...
       ,1),CC((i-1):(i+1),2))...
       /norm(CC(i-1,:)-CC(i,:))...
       /norm(CC(i+1,:)-CC(i,:))...
       /norm(CC(i+1,:)-CC(i-1,:))
end;
```

2. *Curvature estimation via acrlength parameterization.* This is trivial once we figure out how to implement arclength parameterization in MATLAB. The first derivatives $\mathbf{X}'(\lambda)$ are estimated via the finite difference and the second derivative follows similarly. Finite difference for non regular grid is

$$f'(\lambda) = \frac{f(\lambda + \Delta\lambda_1) - f(\lambda)}{\Delta\lambda_1}$$

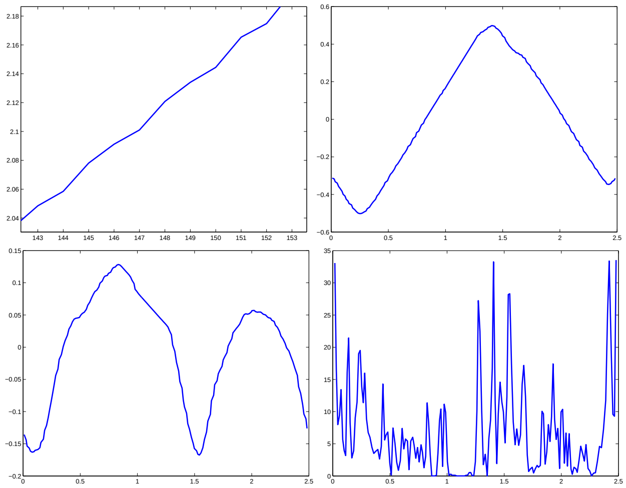

Figure 2: Top left: arc length function λ given as a function of the vertex index. Top right: arc length parameterization of x-component. Bottom left: arc length parameterization of y-component. Bottom right: curvature estimation via arclength parameterization.

and

$$f''(\lambda) = \frac{\frac{f(\lambda+\Delta\lambda_1)-f(\lambda)}{\Delta\lambda_1} - \frac{f(\lambda)-f(\lambda-\Delta\lambda_2)}{\Delta\lambda_2}}{\frac{\Delta\lambda_1+\Delta\lambda_2}{2}}.$$

```
x=CCcurve(:,1);
y=CCcurve(:,2);
dx=diff(x); dy=diff(y);
dl=sqrt(dx.^2 + dy.^2);
%arclength function
for i=1:172
  l(i)=sum(dl(1:i))
end;
%fist difference
Dx=dx./dl(1:172);
figure;plot(l,Dx,'linewidth',2);
Dy=dy./dl(1:172);

%second difference
DDl= (dl(1:171)+dl(2:172))/2;
DDx = diff(Dx)./DDl;
```

```
DDy = diff(Dy)./DDl;
L= (l(1:171)+l(2:172))/2;
kappa=sqrt(DDx.^2 + DDy.^2)
figure;plot(L,kappa,'linewidth',2);
```

3. *Diffusion smoothing on a curve*. Note that $\kappa(0) = \kappa(L)$ where L is the total arclength of $\partial\Omega$. Let us display the curvature function in 2D using the following MATLAB codes.

```
x=CCcurve(:,1) y=CCcurve(:,2)
map=hot(34)
colormap(map)
for i=1:171
  fc=min(ceil(kappa(i)+0.0001),34)
    plot(x(i),y(i),'s',...
     'MarkerEdgeColor','k',...
     'MarkerFaceColor',map(fc,:),...
     'MarkerSize',7)
    hold on;
end; colorbar
```

If we smooth $\kappa(\lambda)$ in interval $[0, \lambda(2\pi))$, we can not grantee the condition $\kappa(0) = \kappa(L)$. How can we smooth periodic data? We can use periodic spline smoothing or diffusion smoothing or iterated kernel smoothing on $\partial\Omega$ directly. Since the diffusion smoothing should gives identical result to the iterated kernel smoothing, we implement iterated kernel smoothing ($\sigma = \sqrt{5} \cdot 0.2 = 0.45$) as follows:

```
kappa=[kappa(171) kappa' kappa(1)]
kernel=inline('exp(-x.^2/sigma^2)...
       /sum(exp(-x.^2/sigma^2))')

for j=1:5
  for i=2:172
    nbr_dist=[dl(i-1) 0 dl(i)];
    weight= kernel(0.2,nbr_dist);
    tem(i-1) = dot(kappa((i-1):(i+1)),...
                   weight);
  end;
  kappa=[tem(171) tem tem(1)]
end;
```

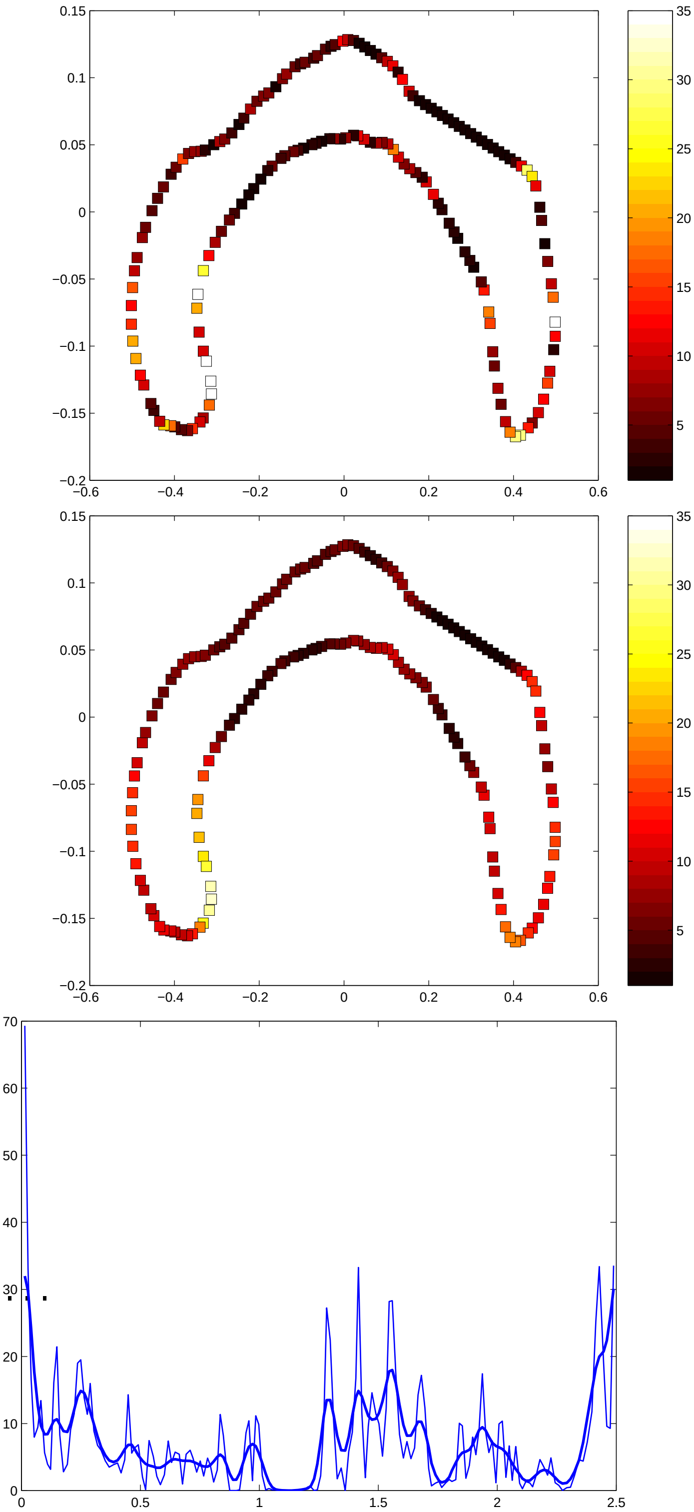

Figure 3: Top: Curvature function κ. Middle: Smoothed curvature function.Bottom: Comparison

Problem 39. Redo **Problem 25** answering following questions. Given parameterization $X(\lambda)$ of curve $\partial\Omega$, what is the 1D Laplacian on $\partial\Omega$? Estimate the Laplacian in our data numerically. Now solve the diffusion equation on $\partial\Omega$ with initial condition κ. Make sure you are not solving the diffusion equation in the parameter space, which is a wrong way to do diffusion smoothing!