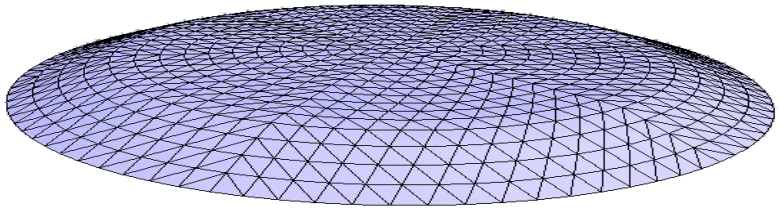

Laplace-Beltrami Eigenstuff Exercises

Martin Reuter – reuter@mit.edu
Mass. General Hospital, Harvard Medical, MIT

Get Files

- Go to http://reuter.mit.edu/tutorial/
- Simply get all.tar.gz (linux, mac) or all.zip (windows)
- Copy and extract files to some place
- Start Matlab and change to that directory

Theory 1D

- Exercise 1
- Plot the Eigenfunctions of Laplace Operator on the Line Segment [0,a]
  - A) For Zero Neumann Boundary condition (derivative zero) plot first 3 functions
  - B) For Zero Dirichlet Boundary condition (function zero) plot first 2 functions
  - Also plot EF 2 vs EF 3 (Neumann) and EF 1 vs EF 2 (Dirichlet)

Matlab code

```
% neumann:
a = 1;
x=[0:0.02:a];
evn1 = ones(length(x)); %constant for n=0
evn2 = cos(1*x*pi/a);  % n=1
evn3 = cos(2*x*pi/a);  % n=2
figure; plot (x,[ evn1; evn2 ; evn3 ] );
figure; hold on ; plot (evn2,evn3,'k.');
plot(evn2,evn3);

% dirichlet:
evd1 = sin(1*x*pi/a);
evd2 = sin(2*x*pi/a);
figure; plot (x,[ evd1; evd2 ] );
figure; hold on ; plot (evd1,evd2,'k.');
plot (evd1,evd2);

% note that -1 *efunc is also an efunc
```

Linear FEM on line

- Exercise 2:
- A) Create a piecewise linear curve (e.g. a spiral) and plot it.
- B) Each line element has two vertices. Create the unit element matrices a and b (2x2) containing the integrals of the local linear form functions (products) on the unit element [0,1]:

$$\varphi_0 = 1 - x \qquad \varphi_1 = x$$

$$a_{ij} = \int_0^1 \varphi_i' \varphi_j' dx \qquad b_{ij} = \int_0^1 \varphi_i \varphi_j dx$$

Matlab code

```
% create spiral
phi = 0:0.2:4*pi;
vnum = length(phi);
x = phi.*cos(phi);
y = phi.*sin(phi);
figure
hold on;
plot(x,y,'k.')
plot(x,y,'b-')

% integrals of local form functions:
a = [1 -1 ; -1 1];
b = [ 1/3 1/6 ; 1/6 1/3 ];
```

Linear FEM on line

- Execise 3
- Construct full matrices A and B (Neumann boundary condition)
  - Use spalloc to allocate sparse matrices of right size (init with zero!)
  - Run through each edge, grab global index of both vertices (left: i, right: j)
  - Compute length of segment (W)
  - For all local index combinations add local results into global matrix A and B (note (k,l) are the local indices and (i,j) the corresponding global indices):

$$A_{ij} = A_{ij} + \frac{1}{W} a_{kl} \qquad B_{ij} = B_{ij} + W\ b_{kl}$$

Matlab code

```
% allocate sparse (filled with zero)
A = spalloc(vnum,vnum,2*vnum);
B = spalloc(vnum,vnum,2*vnum);
for i = 2:vnum
    % global index of first and second vertex of this edge:
    i1 = i-1;     i2 = i;
    % length of segment:
    W = sqrt((x(i2) - x(i1))^2 + (y(i2) -y(i1))^2);
    % fill local stuff into global A matrix
    A(i1,i1) = A(i1,i1) + a(1,1) / W;
    A(i1,i2) = A(i1,i2) + a(1,2) / W;
   % same as line before: symmetric
    A(i2,i1) = A(i2,i1) + a(2,1) / W;
    A(i2,i2) = A(i2,i2) + a(2,2) / W;
    % or simply
    % A([i1,i2],[i1,i2]) = A([i1,i2],[i1,i2]) + (1.0/W) * a;

    B([i1,i2],[i1,i2]) = B([i1,i2],[i1,i2]) + W * b;
end
```

Linear FEM on line

- Exercise 4:
- A) Compute Eigenfunctions and Eigenvalues of this symmetric discrete Operator (smallest 3) using ‘eigs’
  - Reverse the order to have the smallest first
- B) Plot the first non-constant efunc along your curve as color
  - Can be done with surface ([x(:) x(:)], [y(:) y(:)], zeros, colors…)
  - Or wait …

Matlab code

```
% compute efunc evals:
opts.issym = 1;
N = 3; % need the first 2 non zero
[evecsn, evalsn] = eigs(A,B,N,'SM',opts);
% reverse order (smallest first)
evalsn = diag(evalsn);
evalsn = evalsn(end:-1:1)
evecsn = evecsn(:,end:-1:1);

% plot first non-zero efunc as color
figure;
surface([x(:) x(:)],[y(:) y(:)],[zeros(length(x),2)],…
  [ evecsn(:,2) evecsn(:,2)],...
  'FaceColor','none',...
  'EdgeColor','flat',...
  'Marker','none', ...
  'LineWidth',2);
```

Linear FEM on line

- Exercise 5
- Embed curve into spectral domain using Eigenfunction 2 and 3 (first one is constant and useless for Neumann condition)

Matlab code

```
% embedding (onto 2 and 3, as first is constant)

figure
hold on
plot(evecsn(:,2),evecsn(:,3),'k.')
plot(evecsn(:,2),evecsn(:,3),'b-')
```

Linear FEM on line

- Exercise 6:
- Construct A and B for Dirichlet Boundary condition
  - Simply remove row and cols for boundary nodes

Matlab code

```
% dirichlet (remove first and last row and column)
Ad = A(2:end-1,2:end-1);
Bd = B(2:end-1,2:end-1);
% compute efunc evals:
opts.issym = 1;
N = 2;
[evecsd, evalsd] = eigs(Ad,Bd,N,'SM',opts);

% reverse order (smallest first)
evalsd = diag(evalsd);
evalsd = evalsd(end:-1:1)
evecsd = evecsd(:,end:-1:1);
% zero for boundary nodes:
evecsd = [ zeros(1,size(evecsd,2)) ;
                  evecsd ; zeros(1,size(evecsd,2)) ];
```

Linear FEM on line

- Exercise 7
- Embed shape onto first and second Eigenfunction (Dirichlet)
- Embed onto 3rd (Neumann) and 2nd (Dirichlet)
  - What do you expect the result to look like based on the continuous theory?

Matlab code

```
% embedding (onto 1 and 2)
figure
hold on
plot(evecsd(:,1),evecsd(:,2),'k.')
plot(evecsd(:,1),evecsd(:,2),'b-')

% embedding (onto 3neuman and 2dirichlet)
% to create circle:
figure
hold on
plot(evecsn(:,3),evecsd(:,2),'k.')
plot(evecsn(:,3),evecsd(:,2),'b-')
axis equal
```

2D Square Theory

- Exercise 8
- 1. What are the Eigenfunctions on the square of side length a (using Dirichlet Boundary condition)?
- 2. Plot the first 4 Eigenfunctions
  - Use: [X,Y] = meshgrid(x,x) to construct the grid
  - Plot the Eigenfunction Z as a color over the grid using again surface(X,Y,zeros(size(X)),Z, …)
  - Note, first EF is one dimensional, next space has 2 dimensions, and last is again one dimensional (why?).

Matlab code

```
 % square Dirichlet
a = 1; x=[0:0.02:a]; [ X,Y ] = meshgrid(x,x);
% one dim eigenspace
Z = sin(1*X*pi./a) .* sin(1*Y*pi./a);
surface(X,Y,zeros(size(X)), Z,...
  'FaceColor','interp','EdgeColor','none','Marker','none'); axis square

% 2dim eigenspace space
Z = sin(2*X*pi./a) .* sin(1*Y*pi./a);
surface(X,Y,zeros(size(X)), Z,…
  'FaceColor','interp','EdgeColor','none','Marker','none'); axis square

Z = sin(1*X*pi./a) .* sin(2*Y*pi./a);
surface(X,Y,zeros(size(X)), Z,...
  'FaceColor','interp','EdgeColor','none','Marker','none'); axis square

% one dim eigenspace:
Z = sin(2*X*pi./a) .* sin(2*Y*pi./a);
surface(X,Y,zeros(size(X)), Z,...
  'FaceColor','interp','EdgeColor','none','Marker','none'); axis square
```

Finally: Meshes ☺

- Exercise 9:
- Open a mesh: [v,t] = OFF_Import('filename.off')
  - v: is the matrix of vertix coordinates
  - t: is the indices for each triangle (3 corners)
- Compute the bounding box of the shape
- Plot the shape using trisurf
- Compute the maximal vertex degree
- Check if mesh is closed (and if it is manifold)

Matlab code

```
[v,t] = OFF_Import(‘putnamehere.off’);
% bbox:
cmin = min(v)
cmax = max(v)

%plot mesh (color height function):
h = trisurf(t,v(:,1),v(:,2),v(:,3),…
        'EdgeColor','none','FaceColor','interp') ;
Light('Position',[10 0 0],'Style','infinite')
light('Position',[-10 0 0],'Style','infinite')
lighting phong
axis tight equal off
zoom(1)

% vector with vertex degrees:
vdeg = accumarray(t(:),1);
maxdeg = max(vdeg)

is_closed(v,t) % also checks if manifold, look at source
```

Finally: Meshes ☺

- Exercise 10:
- Construct matrices A and B (closed mesh, or Neumann, linear FEM)
  - To speed things up, use: [A,B] = computeAB(v,t)
- Solve sparse symmetric generalized Eigenvalue problem
  - Same as before, this time compute 800 Eigenfunctions (takes a while), if too slow try 500
  - Also don’t forget to reverse order, so that smallest is first

Matlab code

```
% construct matrices (for closed meshes or Neumann
Bcond):
[A , B ] = computeAB(v,t);

% solve sparse symmetric generalized eigenvalue problem:
opts.issym = 1;
N = 800; % will take a while
[evecs, evals] = eigs(A,B,N,'SM',opts);
% evecs are normalized with respect to B
% evecs' * B * evecs = id
% why with respect to B?

% reverse order (smallest first)
evals = diag(evals);
evals = evals(end:-1:1)
evecs = evecs(:,end:-1:1);
```

Finally: Meshes ☺

- Exercise 11:
- Plot the Eigenvalues
- Plot the first few non constant Eigenfunctions (2,3,4,..)
- What do you notice about EF2 (Fiedler Vector)?
- Project the shape into 3D spectral domain
  - Map each vertex to coordinates given by the three EF 2,3,4
  - Plot the mapped shape

Matlab code

```
% plot eigenvalues
plot((1:N),evals,'k-')

% plot first non-constant eigenfunction:
trisurf(t,v(:,1),v(:,2),v(:,3),evecs(:,2),…
            'EdgeColor','none','FaceColor','interp');
light; lighting phong; axis tight equal off; zoom(1)

% same with:
trisurf(t,v(:,1),v(:,2),v(:,3),evecs(:,3),…
             'EdgeColor','none','FaceColor','interp');
trisurf(t,v(:,1),v(:,2),v(:,3),evecs(:,4),…
             'EdgeColor','none','FaceColor','interp');

% project into 3d spectral domain:
trisurf(t,evecs(:,2),evecs(:,3),evecs(:,4),evecs(:,2),…
            'EdgeColor','none','FaceColor','interp');
light; lighting phong ; axis tight equal off ;  zoom(1)
```

Finally: Meshes ☺

- Exercise 12:
- Add Gaussian noise (sigma 0.7) to the right hemisphere
  - Right hemispheres are vertices where x-coord > 0
  - Use get_normals to compute vertex normals
  - Add noise by displacing each vertex along it’s normal

Matlab code

```
% add noise to right hemisphere:
[tn, vn] = get_normals(v,t);
pos = find(v(:,1)>0);
sigma = 0.7;
vnoise = v;
vnoise(pos,:) = v(pos,:) + vn(pos,:) .*…
                (normrnd(0,sigma,size(pos,1),1) * ones(1,3));

trisurf(t,vnoise(:,1),vnoise(:,2),vnoise(:,3),…
      0.5*ones(size(vnoise,1),1),…
      'EdgeColor','none','FaceColor','flat'); axis equal
light('Position',[10 0 0],'Style','infinite')
light('Position',[-10 0 0],'Style','infinite')
lighting flat
material dull
zoom(1)
axis tight equal off
```

Finally: Meshes ☺

- Exercise 13:
- Filter Geometry by cropping:
- Representing the geometry as a linear combination of eigenvectors (use inner product B for projection):

$$c_i = \int_M v f_i d\sigma = \vec{v}' B \vec{f}_i$$

- Compute the linear combination using the first N functions and plot result

$$\vec{v}_{new} = \sum_1^N c_i \vec{f}_i$$

  - Needs to be done for each coordinate
  - N=800 recovers most of the geometry,
  - by removing all larger Eigenfunctions, we basically remove high frequencies (noise etc.)

Matlab code

```
% filter geometry
Nmax = 800;
% represent vnoise as lin comb of eigenvectors
% by projecting xyz coords onto eigenbasis (inner product)
c = (vnoise' * B)* evecs(:,1:Nmax);
% compute linear combination
vsmooth = evecs(:,1:Nmax) * c';

trisurf(t,vsmooth(:,1),vsmooth(:,2),vsmooth(:,3),…
        'EdgeColor','none','FaceColor','flat'); axis equal
light('Position',[10 0 0],'Style','infinite')
light('Position',[-10 0 0],'Style','infinite')
lighting flat
axis tight equal off
zoom(1)
material dull
```

Finally: Meshes ☺

- Exercise 14:
- Filter Geometry with heat kernel:
- Heat kernel weights (use e.g. t = 0.5): $\mathrm{hkw}_i(t) = \mathrm{e}^{-\lambda_i t}$
- Compute and plot the linear combination but modifying the coefficients with the weights:

$$\vec{v}_{new} = \sum_{1}^{N} \mathrm{hkw}_i c_i \vec{f}_i$$

Matlab code

```
% filter geometry with heat kernel
Nmax = 800;
time = 0.5;
plot((1:Nmax),exp(- evals(1:Nmax) * time),'k-')
hkw=(exp(- evals(1:Nmax) * time) * ones(1,3))';
% represent vnoise as lin comb of eigenvectors
c = (vnoise' * B)* evecs(:,1:Nmax);
% compute filtered linear combination:
vsmooth = evecs(:,1:Nmax) * (c.*hkw)';

trisurf(t,vsmooth(:,1),vsmooth(:,2),vsmooth(:,3),…
   'EdgeColor','none','FaceColor','flat'); axis equal
light('Position',[10 0 0],'Style','infinite')
light('Position',[-10 0 0],'Style','infinite')
lighting flat
axis tight equal off
zoom(1)
material dull
```

Finally: Meshes ☺

- Exercise 15:
- Plot Heat Kernel as color on shape with a fixed vertex y
  - Use e.g. time = 400
  - Fix y at gorilla's left foot vertex id: 7570
  - Heat Kernel from Eigensolution (fixed y, free x):

$$K(\vec{x}, \vec{y}, t) = \sum_{n=0}^{\infty} \exp^{-\lambda_n t} f_n(\vec{x}) f_n(\vec{y})$$

  - Other vertices: head 932, right foot 6719

Matlab code

```
% now let's do hot stuff, eg, plot heat kernel at specific
locations:
time = 400;
head = 932;
rfoot = 6719;
lfoot = 7570;
vfix = lfoot;
h = heatkernel(time,vfix,evecs,evals,Nmax);

trisurf(t,v(:,1),v(:,2),v(:,3),h(:,1),…
   'EdgeColor','none','FaceColor','interp'); axis equal
light('Position',[10 0 0],'Style','infinite')
light('Position',[-10 0 0],'Style','infinite')
lighting phong
axis tight equal off
zoom(1)
material dull
```

Finally: Meshes ☺

- Exercise 16
  - (Are you still really working through these? Wow!):
- Plot Heat Kernel Diagonal at fixed vertex for varying t
  - Fix x at gorilla's left foot vertex id: 7570
  - Heat Kernel from Eigensolution (fixed x):

$$K(\vec{x}, \vec{x}, t) = \sum_{i=1}^{N} \mathrm{e}^{-\lambda_n t} f_i(\vec{x}) \; f_i(\vec{x})$$

  - Gives a single plot for fixed x (different t)
  - Other vertices: head 932, right foot 6719

Matlab code

```
time = [0.5:0.5:100]
x=[1,head,rfoot,lfoot];
hdiag = heatdiag(time,x,evecs,evals,Nmax);

plot (time,hdiag(:,:))
xlabel('time');
ylabel('K(t,x,x)');
title('Heat Kernel Diagonal at different t');
lcells = strread(sprintf('x = %d, ',x), '%s','delimiter',',');
legend(lcells);
```

Finally: Meshes ☺

- Exercise 17
- Compute the trace of the heat kernel
  - Use knowledge about Eigenfunctions

$$Z(t) = \int_M K(\vec{x}, \vec{x}, t) dV(\vec{x}) = \sum_{i=1}^{\infty} \mathrm{e}^{-\lambda_i t} \int_M (f_i(\vec{x}))^2 dV(\vec{x}) = ?$$

- Plot Heat Trace (N=800) varying t

Matlab code

```
%heat trace

time = [0.5:0.5:100]
z = sum(exp(- evals(1:Nmax) * time));

plot(time,z);
xlabel('time');
ylabel('Heat Trace Z(t)');
title('Heat Trace at different t');
```

Finally: Meshes ☺

- Exercise 18
- Modify Heat Trace (substitute s := sqrt(t)) and plot

$$Y(s) = 4\pi s^2 \sum_{i=1}^{N} e^{-\lambda_i s^2}$$

- Fit a line through Y(s) between 6<s<8
  - Use regress
  - Plot line all the way through 0 <= s <=8
- Compute surface are of mesh and compare with zero intercept
  - should be close, but since we use only linear FEM, not very accurate.

Matlab code

```
% compute 4pi s^2 sum exp(-lambda_i s^2)
s = sqrt(time);
y = 4*pi*s.^2 .* sum(exp(-evals(1:Nmax) * s.^2));

plot(s,y);
xlabel('sqrt(t)');
ylabel('4 pi t Z(t)');
title('Modified Heat Trace');

% Regression in linear interval to extrapolate
interval = find(s>6 & s<8) ;
X = [ s(interval) ; ones(size(interval)) ]';
B = regress(y(interval)', X);
B(2)
hold on;
plot([0,s],[0,s]*B(1)+B(2),':r');

get_area(v,t)
```

Heat Trace Expansion

- Background about what we just did:
- Heat Trace has an asymptotic expansion (when t->0+)

$$Z(t) = (4\pi t)^{dim(M)/2} \left(\sum_{i=0}^{n} c_i t^{i/2} + O(t^{(n+1)/2}) \right)$$

- Coefficients represent geometric information
- c_0 is the Riemannian volume of M (for surface, the surface area), we just computed c_0.
- Other coefficients contain information about:
  - number of holes, Euler Characteristic, Boundary length

Congrats, you are done …

- If you still have some time
- try to understand all provided code,
- especially computeAB for linear FEM on meshes (see also Talk Part II for details).