Specral Graph Theory and its Applications	September 2, 2004

Lecture 1

Lecturer: Daniel A. Spielman

1.1 A quick introduction

First of all, please call me "Dan". If such informality makes you uncomfortable, you can try "Professor Dan". Failing that, I will also answer to "Prof. Spielman".

As the title suggests, this course is about the eigenvalues and eigenvectors of matrices associated with graphs, and their applications. To be honest, I should say that we will occasionally consider the spectra of matrices that do not come from graphs. But, our approach to these matrices will be sufficiently combinatorial that it will be a lot like looking at graphs. In this lecture, I hope to survey much of the material that we will cover in the course. You will see that I have way too much material in mind to cover in one course. So, we will pare down the list of topics as the semester progresses.

1.2 Matrices for Graphs

First, we recall that a graph $G = (V, E)$ is specified by its vertex set, V, and edge set E. In an undirected graph, the edge set is a set of unordered pairs of vertices. Unless otherwise specified, all graphs will be undirected, and finite.

Without loss of generality, we may assume that $V = \{1, \ldots, n\}$. The most natural matrix to associate with G is its adjacency matrix, A_G, whose entries $a_{i,j}$ are given by

$$a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

I usually prefer to consider the Laplacian matrix of a graph, L_G, whose entries $l_{i,j}$ are given by

$$l_{i,j} = \begin{cases} -1 & \text{if } (i,j) \in E \\ d_i & \text{if } i = j \text{, and} \\ 0 & \text{otherwise,} \end{cases}$$

where d_i is the degree of vertex i.

It is probably not obvious that the eigenvalues or eigenvectors of A_G or L_G should tell you anything useful about the graph G. However, they say a lot! In fact, whenever I meet a strange graph, the first thing I do is examine its eigenvectors and eigenvalues.

Let's quickly compute the spectra of L_G for a simple graph G. In this case, we'll take $V = \{1, 2, 3\}$ and $E = \{(1, 2), (2, 3)\}$:

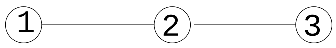


Figure 1.1: The path graph on 3 vertices

We first observe that the vector $(1, 1, 1)$ is an eigenvector of eigenvalue 0. We'll just compute the value we get at the second node when we multiply this vector by L_G. First, we get 2 times the value that was there, 1. We then subtract off the value at each of its neighbors, also 1. Thus, we get 0.

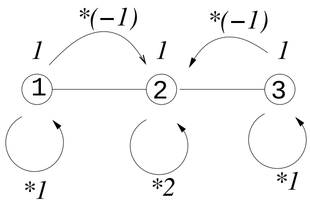


Figure 1.2: The all 1s vector is an eigenvector with eigenvalue 0.

In the figure below, we display the other eigenvectors above the graph, and the vector you get by multiplying by L_G below.

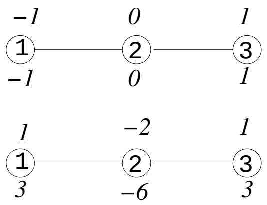


Figure 1.3: The other eigenvectors: $(-1, 0, 1)$ and $(1, -2, 1)$.

In fact, for every Laplacian the all 1s vector is an eigenvector of eigenvalue 0, and all other eigenvalues are non-negative. We'll prove that next class. You might want to think of your own proof before then.

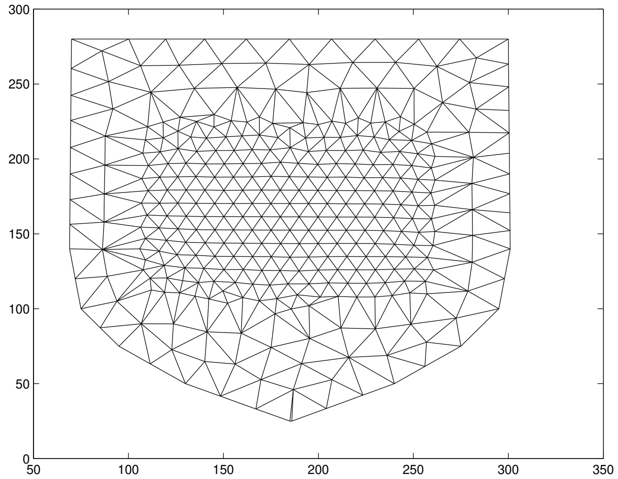
Perhaps the most important thing to take from this example is that an eigenvector is a function on the vertices. That is, it assigns a real number to each vertex.

1.3 Spectral Embeddings

As we just said, an eigenvector assigns a real number to each vertex. So, if we take two eigenvectors, then we obtain two real numbers for each vertex. This suggests a way of drawing a graph: take two eigenvectors, v and w, and plot vertex i at the point $v(i), w(i)$. If we take v and w to be the eigenvectors corresponding to the two smallest non-zero eigenvalues of the Laplacian, this often gives us a very good picture of a graph. Let's look at some examples.

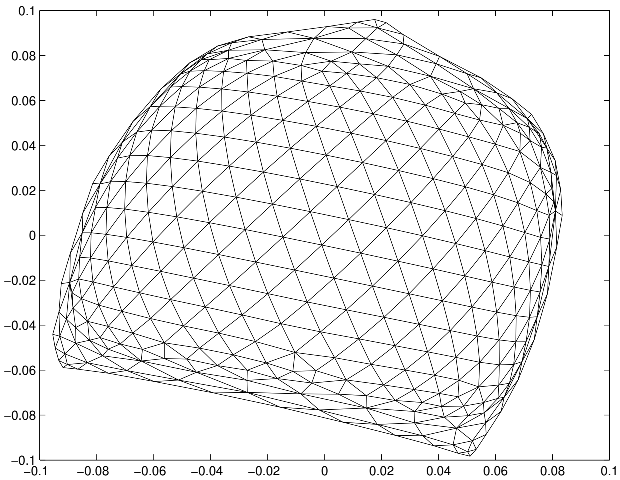
Rather than give you the graph by a list of edges, I'll give you a drawing of it. In the figure, each line segment in the picture is an edge, and each point at which edges meet is a vertex. Clearly, this is a planar graph.

```
load yale
gplot(A,p)
```



In the following figure, we draw the graph using the eigenvectors of the two smallest non-zero eigenvalues of the Laplacian matrix of the graph as the x and y coordinates.

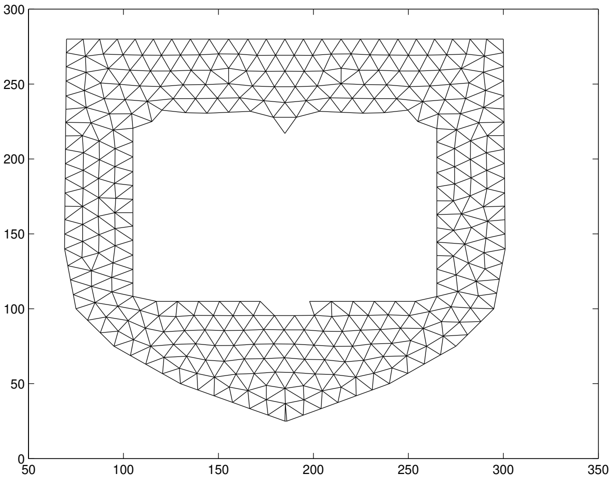
```
gplot(A,x(:,2:3))
```



This picture should convince you that the eigenvectors say a lot about a graph! In particular, for nice planar graphs, they provide a good set of coordinates for the vertices.

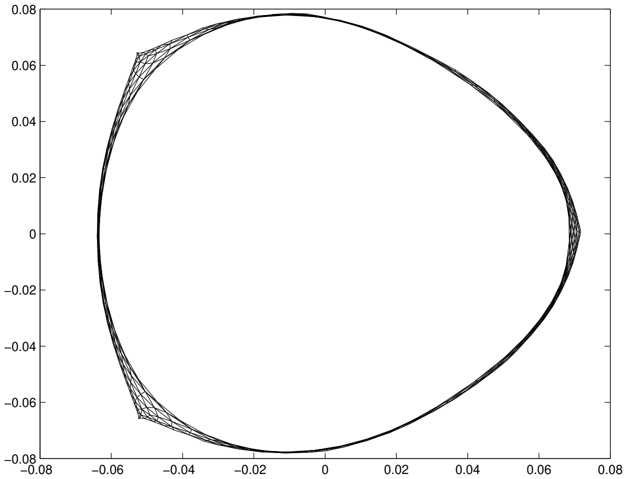
Let's do some more examples. Just to show you that these pictures aren't always ideal, let's take out the book in the center of the shield (and slightly modify the graph).

```
load yale2
gplot(A,p)
```

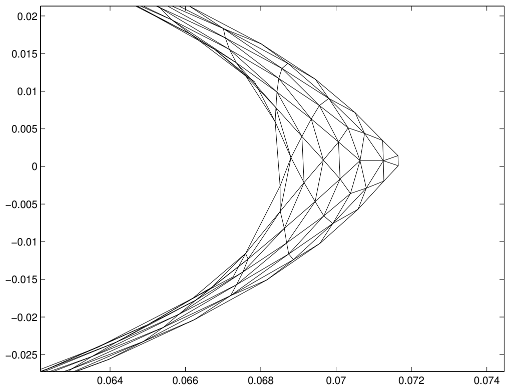


And now, we'll give its spectral embedding.

```
gplot(A,x(:,2:3))
```

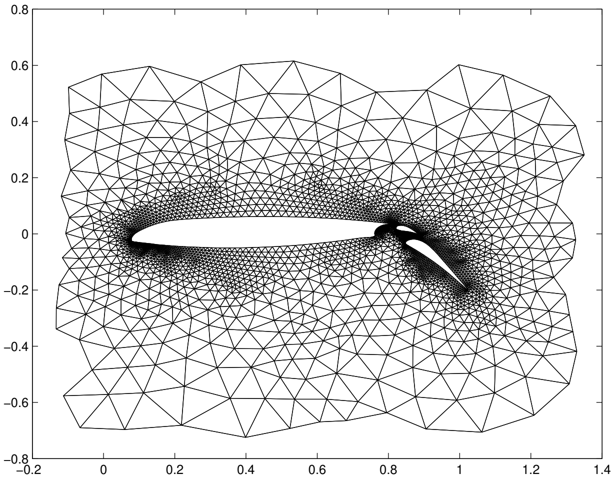


That figure wasn't quite as nice. But, let's zoom in on a little part on the right.



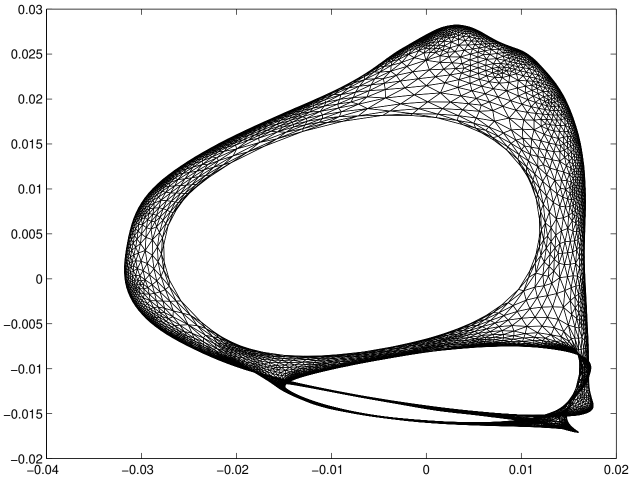
Here's the famous airfoil graph.

```
load airfoil1
gplot(A,xy)
```



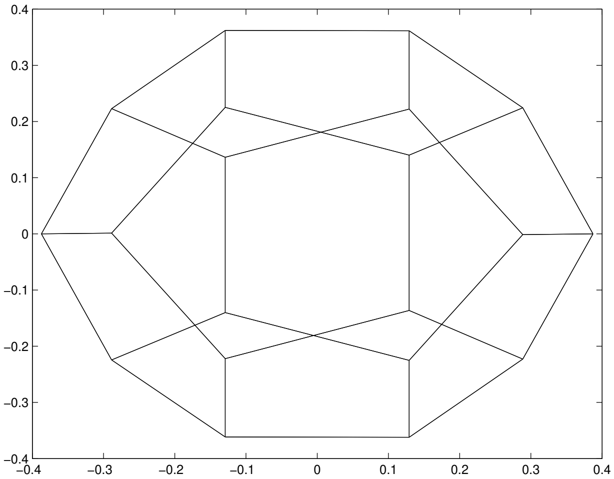
And, here's its spectral embedding.

```
gplot(A,v(:,1:2))
```



Finally, let's look at a spectral embedding of the edges of the dodecahedron.

```
load dodec.txt
E = dodec;
A = sparse(E(:,1),E(:,2),1);
A = A + A';
L = diag(sum(A)) - A;
[v,d] = eig(full(L));
gplot(A,v(:,2:3))
```



You will notice that this looks like what you would get if you squashed the dodecahedron down to the plane. The reason is that we really shouldn't be drawing this picture in two dimensions: the smallest non-zero eigenvalue of the Laplacian has multiplicity three. So, we can't reasonably choose just two eigenvectors. We should be choosing three that span the eigenspace. If we do, we would get the canonical representation of the dodecahedron in three dimensions.

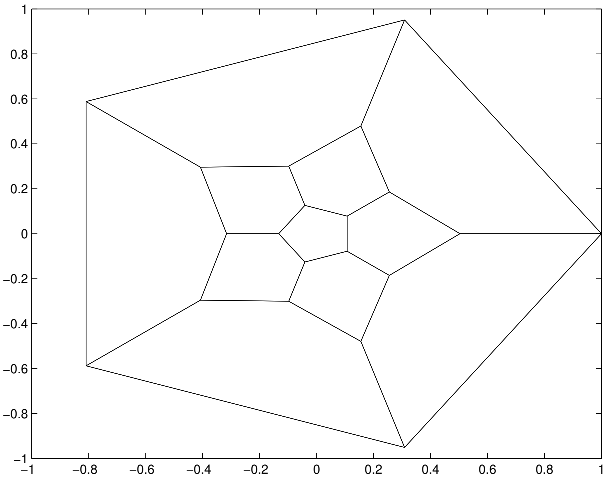
1.3.1 Why?

You might wonder why the eigenvectors supply such good pictures of nice planar graphs. I don't think that anyone has proven a satisfactory theorem about this. But, I do have some intution. For now, I will just mention the intution that comes from a theorem of Tutte. It says that if you take a three-connected planar graph, select any face, fix the positions of the vertices of that face at the corners of a polygon, and then fix every other vertex to be the center of gravity of its neighbors, then you get a straight-line planar embedding. Alternatively, you can think of every edge as a rubber band. We then nail down the vertices on the outside face, and let all the others go where they will. If the graph is planar and three-connected, then this is a planar embedding.

Algebraically, we compute this embedding by solving a linear system in the Laplacian.

Here's an example of what happens when we do this for the dodecahedron.

```
l2 = L;
l2(1:5,:) = 0;
l2(1:5,1:5) = eye(5);
phi = 2*pi*[0:4]/5;
b = [[cos(phi)',sin(phi)'];
      zeros(15,2)];
x = l2 / b;
gplot(A,x)
```



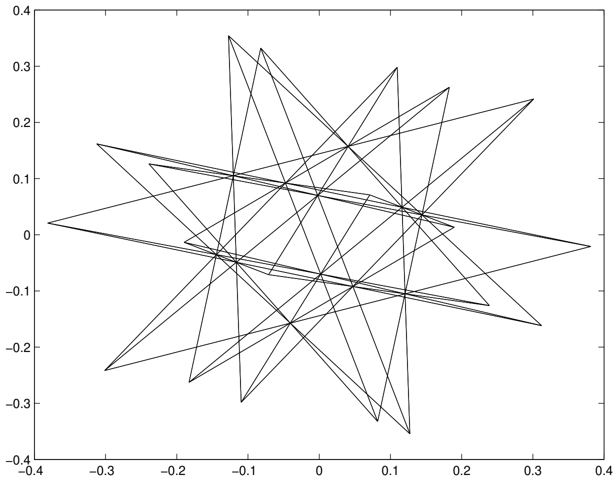
Of course, in the spectral case there is no distinguished outside face, so this isn't exactly what is going on.

I'd be interested in trying to prove something about these embeddings, and would be happy to talk about the problem.

1.3.2 Other eigenvectors

What if we chose to draw using other eigenvectors? For example, we could use the eigenvectors of largest eigenvalue. In the case of the dodecahedron, we get the following embedding from the 19th and 20th eigenvectors:

```
gplot(A,v(:,19:20))
```



In this image, each vertex is far from its neighbors. This suggests that we might be able to use these eigenvectors for coloring. I recall that the problem of graph coloring is that of assigning a color to each vertex so that each edge connects vertices of different colors. The goal is to use as few colors as possible. While we can't always use these eigenvectors for coloring, they do provide a very good heuristic. We might prove later in the semester that they work well on random graphs chosen so that they have colorings with few colors.

1.4 Isomorphism

One of the oldest problems in graph theory is that of determining whether or not two graphs are isomorphic. Two graphs $G = (V, E)$ and $H = (V, F)$ are isomorphic if there is a permutation $\pi : V \to V$ such that $(i, j) \in E$ if and only if $(\pi(i), \pi(j)) \in F$. That is, π is a way of re-labeling the vertices that makes the two graphs the same.

Determining the complexity of graph isomorphism is a maddening task. For every pair of graphs I have ever seen, it has been easy to tell whether or not they are isomorphic. However, the best bound on the worst-case complexity for the problem is $2^{O(\sqrt{n})}$.

From the examples above, one would suspect that eigenvectors and eigenvalues would be very helpful in determining whether or not two graphs are isomorphic. For example, if we let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of L_G and $\mu_1 \leq \mu_2 \leq \cdots \mu_n$ be the eigenvalues of L_H, then $\lambda_i \neq \mu_i$ implies that G and H are non-isomorphic.

Exercise Prove this! Also prove that G and H are isomorphic if and only if there exists a permutation matrix P such that $A_H = P A_G P^T$.

However, this does not determine graph isomorphism because there are non-isomorphic graphs with the same spectra.

But, we can also use the eigenvectors to help test for isomorphism. For example, the spectral embeddings above were uniquely determined up to a rotation. One can prove (and you will)

Exercise Let $\lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of A_G. Assume that λ_i is isolated, that is $\lambda_{i-1} < \lambda_i < \lambda_{i+1}$ and let v_i be the corresponding eigenvector. Let H be a graph isomorphic to G, and let w_i be the ith eigenvector of G. Then, there exists a permutation π such that $v_i(j) = w_i(\pi(j))$.

So, if we can find a few eigenvectors that map each vertex to a distinct point, then we can use them to test for isomorphism.

Unfortuntely, there exist graphs for which the eigenvectors do not tell us very much.

The following graph has an eigenvector that only has two values.

```
load gcube;
[v,d] = eig(full(m));
v(:,1)

ans =

   -0.2500
   -0.2500
    0.2500
    0.2500
    0.2500
    0.2500
   -0.2500
   -0.2500
    0.2500
    0.2500
   -0.2500
   -0.2500
   -0.2500
   -0.2500
    0.2500
    0.2500
```

However, this is not fatal. As we will see during the course, if every eigenvector has multiplicity less than m, then it is possible to test isomorphism in time $n^{m+O(1)}$.

However, there exist graphs in which there are no non-trivial eigenvalues of small multiplicity.

```
load lsg;
```

```
[v,d] = eig (full (A));
diag (d)


ans =

   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
   -3.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
    2.0000
   12.0000
```

Well, there was one eigenvector of multiplicity one, but it corresponded to the all 1s eigenvector.

Graphs like these usually come from algebraic structures. For example, the first was a Cayley graph. I'll describe general Cayley graphs later, but I'll tell you now where this one came from. Each vertex was identified with an element of $\{0,1\}^4$. I then chose 6 special elements called the generators of the graph,

$$g_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad g_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad g_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$g_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad g_5 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad g_6 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

I then created an edge $(x, x + g_i)$ for each $x \in \{0,1\}^4$ and $1 \leq i \leq 6$.

Cayley graphs are generalizations of this construction, in which we replace $\{0,1\}^4$ with any group and require that the generator set be closed under inversion. The eigenvectors and eigenvalues of Cayley graphs are closely related to their irreducible representations. When the group is abelian, the eigenvectors do not depend on the choice of generator set.

The second graph was a strongly-regular graph. These are a maddening family of graphs in which the eigenvectors essentially provide no help in determining isomorphism.

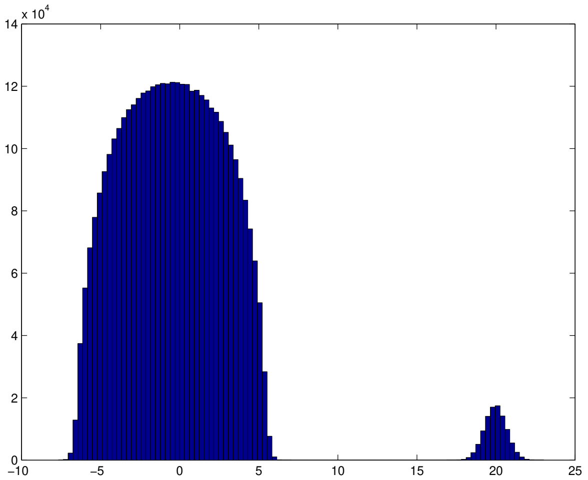
Algebraic constructions of graphs provide the power behind many algorithms and constructions in the theory of computing and error-correcting codes. We will see many of them.

1.5 Random Graphs

There are two fancy ways to construct graphs: by algebraic constructions and at random. It turns out that random graphs are almost as nice as algebraic graphs, and are sometimes nicer. If they are large enough, random graphs should really be thought of as a special family of graphs. We will see that the spectra of random graphs are very well behaved.

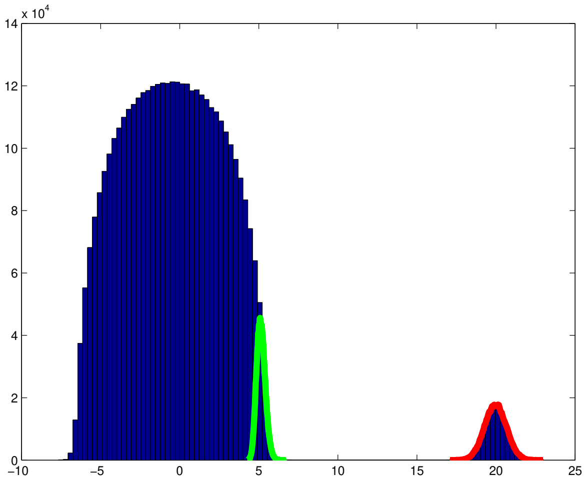
The most natural random graph is one in which each edge is chosen to exist with probability 1/2, independently from all the others. We will consider the adjacency matrix of such a graph, and the density function of its eigenvalues. To generate this density, we will compute a histogram. That is, we will generate many adjacency matrices, compute their eigenvalues, and then histogram all the eigenvalues (lumping them all together). Here's the matlab code.

```
d = 40;
its = 100000;
E = zeros(d,its);
for i = 1:its,
a = rand(d) > .5;
a = triu(a,1);
a = a + a';
e = sort(eig(a));
E(:,i) = e;
end
[y,x] = hist(E(:),100);
hist(E(:),100);
```



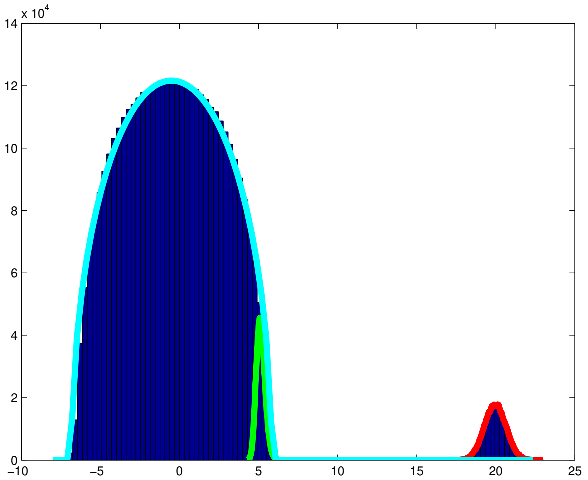
Now, I'll add in histograms of just the largest and second-to-largest eigenvalues.

```
[y1,x1] = hist(E(40,:),100);
sc = x(2) - x(1);
sc1 = x1(2) - x1(1);
hold on
plot(x1,y1 *sc / sc1, 'r',...
   'LineWidth',5)
[y2,x2] = hist(E(39,:),100);
sc2 = x2(2) - x2(1);
plot(x2,y2 *sc / sc2, 'g',...
   'LineWidth',5)
```



Note that these distributions are very tightly concentrated. Almost everything that we are seeing here can be proved analytically. This includes the form of the big hump in the limit.

```
x1 = x / sqrt(d);
s1 = sqrt(1-x1.^2);
z = s1 / s1(23) * y(23);
plot(x-1/2,z,'c','LineWidth',5)
```



1.6 The Spectral Gap

We now get to my favorite eigenvalue, λ_2 of the Laplacian. That is, the smallest non-zero eigenvalue. It is often called the Fiedler value. Whenever I see a graph, the first thing that I want to know about it is its Fiedler value. In particular, it helps me distinguish many families of graphs. Here are some sample values

Graph	Fiedler Value
Path	$\Theta(1/n^2)$
Grid	$\Theta(1/n)$
3d grid	$\Theta(n^{-2/3})$
Expander	$\Theta(1)$
binary tree	$\Theta(1/n)$
dumbell	$\Theta(1/n)$

By a dumbell, I mean two expanders joined by one edge. If you don't yet know what an expander is, don't worry. You will.

One of the reasons the Fiedler value is exciting is that it tells you how well you can cut a graph. A cut of a graph is a division of its vertices into two sets, S and $\bar{S}$. We usually want to find cuts that cut as few edges as possible. We let $E(S, \bar{S})$ denote the set of edges whose vertices lie on opposite sides of the cut. We then define the *ratio* of the cut to be

$$\phi(S) = \frac{\left|E(S, \bar{S})\right|}{\min(|S|, |\bar{S}|)}.$$

The best cut is the one of minimum ratio, and its quality is the *isoperimetric number* of a graph

$$\phi(G) = \min_S \phi(S).$$

A version of Cheeger's inequality says that the isoperimetric number is intimately related to λ_2.

Theorem 1.6.1.

$$\phi \geq \lambda_2 \geq \frac{\phi^2}{2d},$$

where d is an upper bound the the degree of every vertex in the graph.

If we appropriately scale the isoperimetric number and the Laplacian, it is possible to get rid of the d in the inequality.

By Cheeger's inequality, λ_2 says something good about every graph. If λ_2 is small, then it is possible to cut the graph into two pieces without cutting too many edges. If λ_2 is large, then every cut of the graph must cut many edges. Expanders are graphs in which λ_2 is large, where large might mean $\Theta(1)$ or very close to d. I could teach a whole course on expanders, and some have. The short story is that good constructions of expanders exist, and they enable good constructions of all sort of other things, from error-correcting codes to pseudo-random generators.

Here's another useful property of expanders.

Theorem 1.6.2. *Let $G = (V, E)$ be a d-regular connected graph on n nodes such that $d - \lambda_2 < \mu$. Then, for every $S, T \subseteq V$ with $|S| = \alpha n$ and $|T| = \beta n$, we have*

$$|E(S, T) - d\alpha\beta n| \leq \mu n\sqrt{(\alpha - \alpha^2)(\beta - \beta^2)}.$$

That is, if μ is small, then for every pair of sets of vertices S and T, the number of edges between S and T is close to what you would expect to find if you chose S and T at random. There are explicit constructions of expanders in which $\mu \sum 2\sqrt{d-1}$.

Using λ_2, we can also answer questions such as: "how much does a random subgraph of G look like G"?

1.7 Quantum Computing

1.8 Graph approximation and preconditioning

What does it mean for one graph to approximate another? Here is one definition.

Definition 1.8.1. *A graph G is a c-approximation of H if for all $x \in R^n$,*

$$x^T L_G x \leq x^T L_H x \leq x^T L_G x.$$

In the class, we will see how to prove the following two theorems.

Theorem 1.8.2. *For every weighted graph G, there exists a weighted graph H with at most $n \log^{O(1)} n$ edges that $(1+\epsilon)$ approximates G.*

Theorem 1.8.3. *For every weighted graph G, there exists a weighted graph H with at most $n + t \log^{O(1)} n$ edges that n/t approximates G.*

We will see that these results can be used to quickly solve diagonally-dominant linear systems.

1.9 Summary

I see three ways of looking at spectral graph theory: descriptive, algorithmic, and controlling. In the descriptive world, we determine what we can learn about a graph from its spectra. In the algorithmic framework, we use this descriptive knowledge to design algorithms. In the controlling framework, we design graphs whose eigenvalues meet certain restrictions. If believe that by improving our descriptive understanding, we can improve efforts at control.

1.10 Mechanics

This course is designed for graduate students. While there is no book for the course, I will supply reference material for each lecture. Sometimes this will be a paper, and often it will be my lecture notes.

The work load should not be too high. There will be two problem sets, and each student will scribe one lecture. If there are few enough students, then each may give a presentation at the end of the semester.