Spectral Graph Theory Lecture 1

Introduction

Daniel A. Spielman August 29, 2018

1.1 First Things

1. Please call me "Dan". If such informality makes you uncomfortable, you can try "Professor Dan". If that fails, I will also answer to "Prof. Spielman".
2. If you are going to take this course, please sign up for it on Canvas. This is the only way you will get emails like "Problem 3 was false, so you don't have to solve it".
3. This class meets this coming Friday, September 31, but not on Labor Day, which is Monday, September 3.

1.2 Introduction

I have three objectives in this lecture: to give you an overview of the material we will cover this semester, to help you decide if this course is right for you, and to tell you how the course will work.

As the title suggests, this course is about the eigenvalues and eigenvectors of matrices associated with graphs, and their applications. I will never forget my amazement at learning that combinatorial properties of graphs could be revealed by an examination of the eigenvalues and eigenvectors of their associated matrices. I hope to both convey my amazement to you and to make it feel like common sense. I'm now shocked when any important property of a graph is not revealed by its eigenvalues and eigenvectors.

This class will fundamentally be a math class, and my emphasis is on material that I find beautiful and/or useful. I'll present a lot of theorems, a few algorithms, and a bunch of open problems.

1.3 Mechanics

There is no book for this course, but I will produce notes for every lecture. **You should read the lecture notes.** They will often contain material that I did not have time to cover in class. They will sometimes contain extra expositions of elementary topics. I will try to make the notes available before lecture. Some students will want to print them out for reference during lecture.

Given that I am providing lecture notes, you might not need to take notes during lecture. I, however, take notes during every lecture that I attend. It helps me pay attention and remember what is going on. But, there are many different learning styles. You may prefer to just listen.

If you would like a book that covers some of the material from the course, I suggest one of

> "Algebraic Graph Theory" by Chris Godsil and Gordon Royle,
>
> "Spectral Graph Theory" by Fan Chung, or
>
> "Algebraic Combinatorics" by Chris Godsil.

I expect to produce around 5 or 6 problem sets during the semester. Some of the problems I assign in these will be very hard. You will be allowed to work on them in small groups.

For some lectures, such as today's, I have assigned a number of "exercises" at the end of the lecture notes. You should solve these on your own, as soon after lecture as possible. You should not hand them in. They are just to help you practice the material. Today's exercises are a review of fundamental linear algebra. I will put the solutions to some of them on Canvas.

There will be no tests or exams.

1.3.1 This is a graduate course

As some undergrads are thinking about taking this course, I thought I should explain the main differences between an undergraduate course and a graduate course, and the differences in outlook between undergrads and graduate students.

Graduate school is essentially pass/fail. Graduate students either write a thesis and graduate, or they do not. Their grades in courses do not matter very much. Most are here because they think they might learn something in this course that they will find useful in their careers. This means that some of them will work very hard.

Graduate students are also occasionally occupied with other responsibilities, like teaching and research. For this reason, I will give students at least two weeks to complete the problems I assign. However, I recommend that you solve the easier problems immediately.

Graduate students routinely take courses for which they do not have all the prerequisite knowledge. I assume that they can learn anything elementary as needed. Wikipedia makes this much easier than it used to be.

Finally, graduate courses are not as "user friendly" as undergraduate courses. I make no guarantees about what will happen in this course. I may assign more or fewer problem sets than I have announced. I may completely change the topics that I decide to cover. You have been warned.

1.3.2 Other courses

I have adjusted my selection of material for this course to decrease overlap with others. For example, I am omitting some material that will be included in S&DS 684a: Statistical Inference on Graphs and S&DS 615b: Introduction to Random Matrix Theory.

1.4 Background: Graphs

First, we recall that a graph $G = (V, E)$ is specified by its vertex[1] 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, simple (having no loops or multiple edges) and finite. We will sometimes assign weights to edges. These will usually be real numbers. If no weights have been specified, we view all edges as having weight 1. This is an arbitrary choice, and we should remember that it has an impact.

Graphs (also called "networks") are typically used to model connections or relations between things, where "things" are vertices. However, I often prefer to think of the edges in a graph as being more important than the vertices. In this case, I may just specify an edge set E, and ignore the ambient vertex set.

Common "natural" examples of graphs are:

- Friendship graphs: people are vertices, edges exist between pairs of people who are friends (assuming the relation is symmetric).
- Network graphs: devices, routers and computers are vertices, edges exist between pairs that are connected.
- Circuit graphs: electronic components, such as transistors, are vertices: edges exist between pairs connected by wires.
- Protein-Protein Interaction graphs: proteins are vertices. Edges exist between pairs that interact. These should really have weights indicating the strength and nature of interaction. Most other graphs should to.

It is much easier to study abstract, mathematically defined graphs. For example,

- The path on n vertices. The vertices are $\{1, \dots n\}$. The edges are $(i, i+1)$ for $1 \leq i < n$.
- The ring on n vertices. The vertices are $\{1, \dots n\}$. The edges are all those in the path, plus the edge $(1, n)$.
- The hypercube on 2^k vertices. The vertices are elements of $\{0, 1\}^k$. Edges exist between vertices that differ in only one coordinate.

1.5 Matrices for Graphs

The naive view of a matrix is that it is essentially a spreadsheet—a table we use to organize numbers. This is like saying that a car is an enclosed metal chair with wheels. It says nothing about what it does!

[1] I will use the words "vertex" and "node" interchangeably. Sorry about that.

I will use matrices to do two things. First, I will view a matrix $\boldsymbol{M}$ as providing an function that maps a vector $\boldsymbol{x}$ to the vector $\boldsymbol{M}\boldsymbol{x}$. That is, I view $\boldsymbol{M}$ as an operator. Second, I view a matrix $\boldsymbol{M}$ as providing a function that maps a vector $\boldsymbol{x}$ to a number $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x}$. That is, I use $\boldsymbol{M}$ to define a quadratic form.

1.5.1 A spreadsheet

We will usually write V for the set of vertices of a graph, and let n denote the number of vertices. There are times that we will need to order the vertices and assign numbers to them. In this case, they will usually be $\{1, \ldots, n\}$. For example, if we wish to draw a matrix as a table, then we need to decide which vertex corresponds to which row and column.

The most natural matrix to associate with a graph G is its adjacency matrix[2], $\boldsymbol{M}_G$, whose entries $\boldsymbol{M}_G(a, b)$ are given by

$$\boldsymbol{M}_G(a, b) = \begin{cases} 1 & \text{if } (a, b) \in E \\ 0 & \text{otherwise.} \end{cases}$$

It is important to observe that I index the rows and columns of the matrix by vertices, rather than by number. Almost every statement that we make in this class will remain true under renaming of vertices. The first row of a matrix has no special importance. To understand this better see the exercises at the end of the lecture.

While the adjacency matrix is the most natural matrix to associate with a graph, I also find it the least useful. Eigenvalues and eigenvectors are most meaningful when used to understand a natural operator or a natural quadratic form. The adjacency matrix provides neither.

1.5.2 An operator

The most natural operator associated with a graph G is probably its diffusion operator. This operator describes the diffusion of stuff among the vertices of a graph and how random walks behave. We will save further discussion of this perspective for a later lecture.

1.5.3 A quadratic form

The most natural quadratic form associated with a graph is defined in terms of its Laplacian matrix,

$$\boldsymbol{L}_G \overset{\text{def}}{=} \boldsymbol{D}_G - \boldsymbol{M}_G,$$

where $\boldsymbol{D}_G$ is the diagonal matrix in which $\boldsymbol{D}_G(a, a)$ is the degree of vertex a. We will usually write $\boldsymbol{d}(a)$ for the degree of vertex a. In an unweighted graph, the degree of a vertex is the number of edges attached to it. In the case of a weighted graph, we use the *weighted degree*: the sum of the weights of the edges attached to the vertex a.

[2] I am going to try to always use the letter $\boldsymbol{M}$ for the adjacency matrix, in contrast with my past practice which was to use $\boldsymbol{A}$. I will use letter like a and b to denote vertices.

Given a function on the vertices, $\boldsymbol{x} \in \mathbb{R}^V$, the Laplacian quadratic form is

$$\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} = \sum_{(a,b)\in E} (\boldsymbol{x}(a) - \boldsymbol{x}(b))^2. \tag{1.1}$$

This form measures the smoothness of the function $\boldsymbol{x}$. It will be small if the function $\boldsymbol{x}$ does not jump too much over any edge.

I use the notation $\boldsymbol{x}(a)$ to denote the coordinate of vector $\boldsymbol{x}$ corresponding to vertex a. Other people often use subscripts for this, like $\boldsymbol{x}_a$. I reserve subscripts for other purposes.

1.6 Background: Spectral Theory

I now review the highlights of the spectral theory for symmetric matrices. Almost all of the matrices we consider in this course will be symmetric or will be similar[3] to symmetric matrices.

We recall that a vector $\boldsymbol{\psi}$ is an eigenvector of a matrix $\boldsymbol{M}$ with eigenvalue λ if

$$\boldsymbol{M}\boldsymbol{\psi} = \lambda\boldsymbol{\psi}. \tag{1.2}$$

That is, λ is an eigenvalue if and only if $\lambda\boldsymbol{I} - \boldsymbol{M}$ is a singular matrix. Thus, the eigenvalues are the roots of the characteristic polynomial of $\boldsymbol{M}$:

$$\det(x\boldsymbol{I} - \boldsymbol{M}).$$

Theorem 1.6.1. *[The Spectral Theorem] If $\boldsymbol{M}$ is an n-by-n, real, symmetric matrix, then there exist real numbers $\lambda_1, \ldots, \lambda_n$ and n mutually orthogonal unit vectors $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n$ and such that $\boldsymbol{\psi}_i$ is an eigenvector of $\boldsymbol{M}$ of eigenvalue λ_i, for each i.*

This is the great fact about symmetric matrices. If the matrix is not symmetric, it might not have n eigenvalues. And, even if it has n eigenvalues, their eigenvectors will not be orthogonal[4]. In fact, if $\boldsymbol{M}$ is not symmetric, then its eigenvalues and eigenvalues might be the wrong thing to look at.

I remind you that the eigenvectors are not uniquely determined, although the eigenvalues are. If $\boldsymbol{\psi}$ is an eigenvector, then $-\boldsymbol{\psi}$ is as well. Some eigenvalues can be repeated. If $\lambda_i = \lambda_{i+1}$, then $\boldsymbol{\psi}_i + \boldsymbol{\psi}_{i+1}$ will also be an eigenvector of eigenvalue λ_i. Generally, the eigenvectors of a given eigenvalue are only determined up to an orthogonal transformation.

Fact 1.6.2. *The Laplacian matrix of a graph is positive semidefinite. That is, all its eigenvalues are nonnegative.*

Proof. Let $\boldsymbol{\psi}$ be a unit eigenvector of $\boldsymbol{L}$ of eigenvalue λ. Then,

$$\boldsymbol{\psi}^T \boldsymbol{L}\boldsymbol{\psi} = \boldsymbol{\psi}^T \lambda\boldsymbol{\psi} = \lambda = \sum_{(a,b)\in E} (\boldsymbol{\psi}(a) - \boldsymbol{\psi}(b))^2 > 0.$$

$\square$

[3]A matrix $\boldsymbol{M}$ is similar to a matrix $\boldsymbol{B}$ if there is a non-singular matrix $\boldsymbol{X}$ such that $\boldsymbol{X}^{-1}\boldsymbol{M}\boldsymbol{X} = \boldsymbol{B}$. In this case, $\boldsymbol{M}$ and $\boldsymbol{B}$ have the same eigenvalues. See the exercises at the end of this lecture.

[4]You can prove that if the eigenvectors are orthogonal, then the matrix is symmetric.

We always number the eigenvalues of the Laplacian from smallest to largest. Thus, $\lambda_1 = 0$. We will refer to λ_2, and in general λ_k for small k, as *low-frequency* eigenvalues. λ_n is a *high-frequency* eigenvalue. We will see why in a moment.

1.7 Overview of the course

We will begin the course by learning about the eigenvalues and eigenvectors of many special graphs. These will include simple graphs like paths, rings, stars, trees and hypercubes, and we will eventually get to Cayley graphs and Strongly Regular Graphs.

Before we get to any theorems, I would like to convince you that the eigenvalues and eigenvectors of graphs are meaningful by showing you some examples. I will do these examples in Julia using a Jupyter notebook. I include snippets of the code and the images they generate in this text, and have provided the notebook on the course webpage.

1.7.1 Paths

A path graph has vertices $\{1, \ldots, n\}$ and edges $(i, i+1)$ for $1 \leq i < n$. Here is the adjacency matrix of a path graph on 4 vertices.

```
M = path_graph(4)
Matrix(M)
 0.0  1.0  0.0  0.0
 1.0  0.0  1.0  0.0
 0.0  1.0  0.0  1.0
 0.0  0.0  1.0  0.0
```

And, here is its Laplacian matrix

```
Matrix(lap(M))
  1.0  -1.0   0.0   0.0
 -1.0   2.0  -1.0   0.0
  0.0  -1.0   2.0  -1.0
  0.0   0.0  -1.0   1.0
```

Here are the eigenvalues of a longer path.

```
L = lap(path_graph(10))
E = eigen(Matrix(L))
println(E.values)

[0.0, 0.097887, 0.381966, 0.824429, 1.38197, 2.0, 2.61803, 3.17557, 3.61803, 3.90211]
```

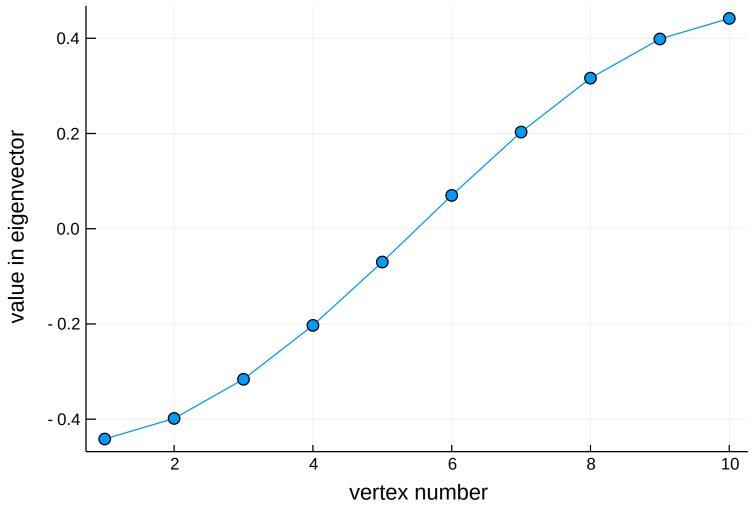
The eigenvector of the zero-eigenvalue is a constant vector (up to numerical issues):

```
E.vectors[:,1]

 0.31622776601683755
 0.31622776601683716
 0.31622776601683766
 0.3162277660168381
 0.31622776601683855
 0.3162277660168381
 0.3162277660168385
 0.31622776601683805
 0.3162277660168378
 0.3162277660168378
```

The eigenvector of λ_2 is the lowest frequency eigenvector, as we can see that it increases monotonically along the path:

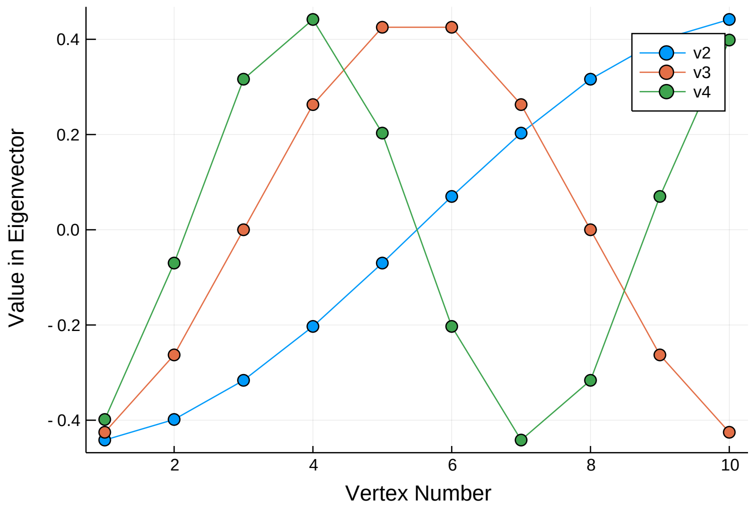
```
v2 = E.vectors[:,2]

 -0.44170765403093937
 -0.39847023129620024
 -0.316227766016838
 -0.20303072371134553
 -0.06995961957075425
  0.06995961957075386
  0.2030307237113457
  0.31622776601683766
  0.3984702312961997
  0.4417076540309382
```

Let's plot that.



```
plot(v2,marker=5,legend=false)
xlabel!("vertex number")
ylabel!("value in eigenvector")
```

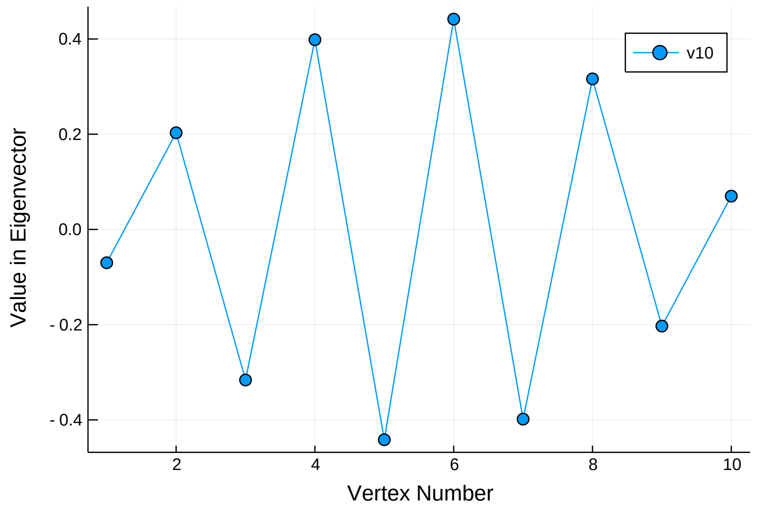
The x-axis is the name/number of the vertex, and the y-axis is the value of the eigenvector at that vertex. Now, let's look at the next few eigenvectors.



```
Plots.plot(E.vectors[:,2],label="v2",marker = 5)
Plots.plot!(E.vectors[:,3],label="v3",marker = 5)
Plots.plot!(E.vectors[:,4],label="v4",marker = 5)
xlabel!("Vertex Number")
ylabel!("Value in Eigenvector")
```

You may now understand why I refer to these as the low-frequency eigenvectors. The curves they trace out resemble the low-frequency modes of vibration of a string. The reason for this is that the path graph can be viewed as a discretization of the string, and its Laplacian matrix is a discretization of the Laplace operator. We will relate the low-frequency eigenvalues to connectivity.

In contrast, the highest frequency eigenvalue alternates positive and negative with every vertex. We will show that these may be related to problems of graph coloring and finding independent sets.



```
Plots.plot(E.vectors[:,10],label="v10",marker=5)
xlabel!("Vertex Number")
ylabel!("Value in Eigenvector")
```

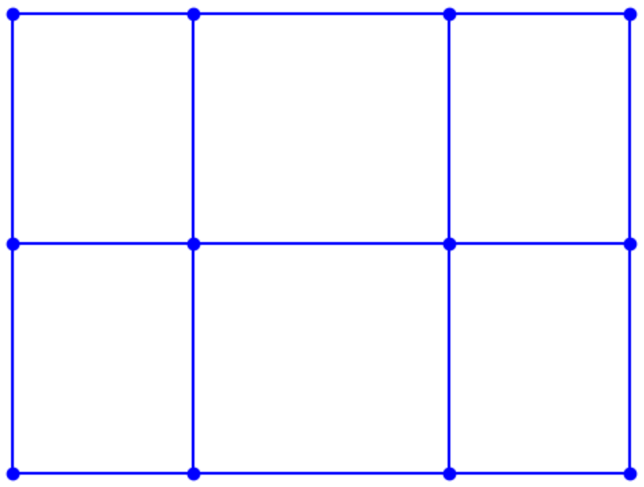
1.7.2 Spectral Graph Drawing

We can often use the low-frequency eigenvalues to obtain a nice drawing of a graph. For example, here is 3-by-4 grid graph, and its first two non-trivial eigenvectors. Looking at them suggests that they might provide nice coordinates for the vertices.

```
M = grid2(3,4)
L = lap(M)
E = eigen(Matrix(L))
V = E.vectors[:,2:3]

 -0.377172   0.353553
 -0.15623    0.353553
  0.15623    0.353553
  0.377172   0.353553
 -0.377172  -1.66533e-16
 -0.15623   -4.16334e-16
  0.15623   -5.82867e-16
  0.377172   2.77556e-16
```

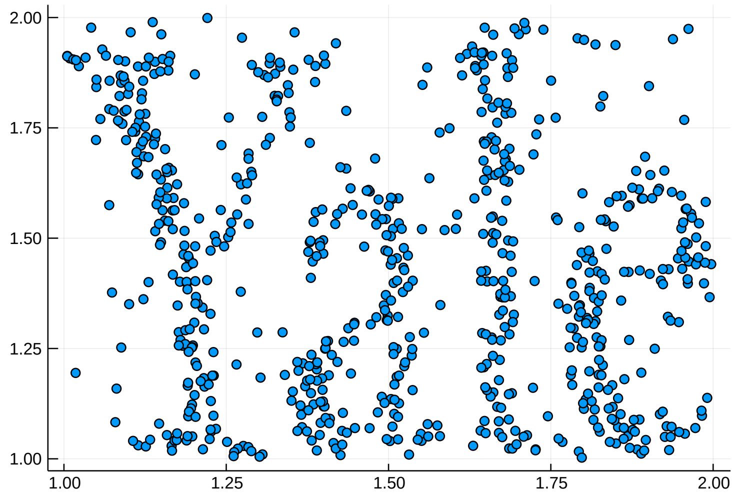
```
-0.377172  -0.353553
-0.15623   -0.353553
 0.15623   -0.353553
 0.377172  -0.353553
```

In the figure below, we use these eigenvectors to draw the graph. Vertex a be been plotted at coordinates $\boldsymbol{\psi}_2(a), \boldsymbol{\psi}_3(a)$. That is, we use $\boldsymbol{\psi}_2$ to provide a horizontal coordinate for every vertex, and $\boldsymbol{\psi}_3$ to obtain a vertical coordinate. We then draw the edges as straight lines.



```
plot_graph(M,V[:,1],V[:,2])
```

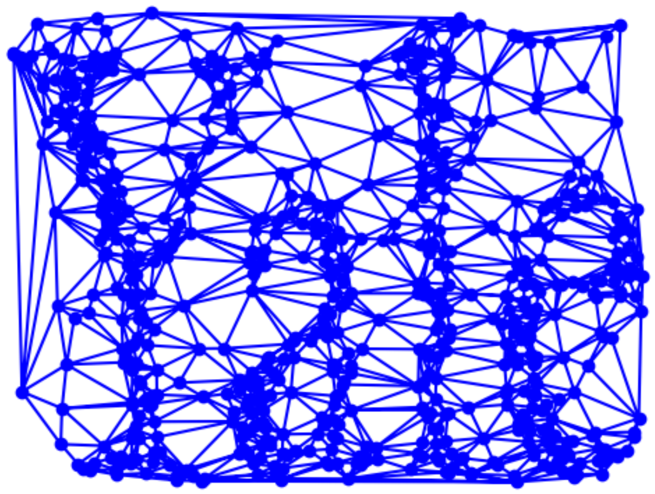
Let's do a fancier example that should convince you something interesting is going on. I begin by generating points by sampling them from the Yale logo.



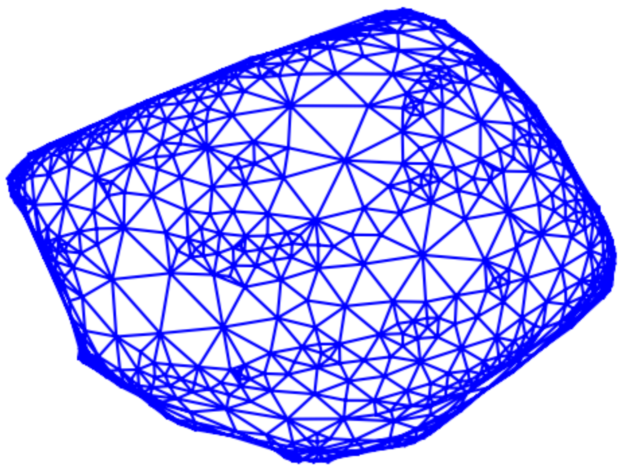
```
@load "yale.jld2"
scatter(xy[:,1],xy[:,2],legend=false)
```

I then construct a graph on them by forming their Delaunay triangulation. I won't get to teach about Delaunay triangulations during this course. But, they are terrific and I recommend that you look them up.

Since the vertices came with coordinates, it was easy to draw a nice picture of the graph. But, what if we just knew the graph, and not the coordinates? We could generate coordinates by computing two eigenvectors, and using each as a coordinate. Below, I plot vertex a at position $\psi_2(a), \psi_3(a)$, and again draw the edges as straight lines.



```
plot_graph(a,xy[:,1],xy[:,2])
```



```
plot_graph(a, v2,v3, dots=false)
```

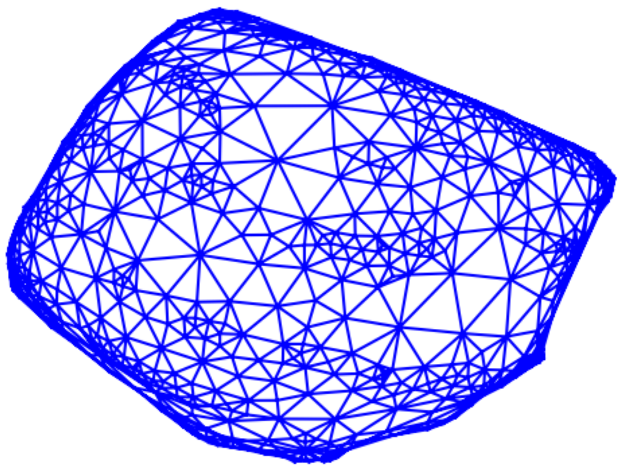
That's a great way to draw a graph if you start out knowing nothing about it. It's the first thing I do whenever I meet a strange graph. Note that the middle of the picture is almost planar, although edges do cross near the boundaries.

1.7.3 Graph Isomorphism

It is important to note that the eigenvalues do not change if we relabel the vertices. Moreover, if we permute the vertices then the eigenvectors are similarly permuted. That is, if $\boldsymbol{P}$ is a permutation matrix, then

$$\boldsymbol{L}\boldsymbol{\psi} = \lambda\boldsymbol{\psi} \quad \text{if and only if} \quad (\boldsymbol{P}\boldsymbol{L}\boldsymbol{P}^T)(\boldsymbol{P}\boldsymbol{\psi}) = \lambda(\boldsymbol{P}\boldsymbol{\psi}),$$

because $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}$. To prove it by experiment, let's randomly permute the vertices, and plot the permuted graph.



```
Random.seed!(1)
p = randperm(size(a,1))
M = a[p,p]
E = eigen(Matrix(lap(M)))
V = E.vectors[:,2:3]
plot_graph(M,V[:,1],V[:,2], dots=false)
```

Note that this picture is slightly different from the previous one: it has flipped vertically. That's because eigenvectors are only determined up to signs, and that's only if they have multiplicity 1. This gives us a very powerful heuristic for testing if one graph is a permutation of another (this is the famous "Graph Isomorphism Testing Problem"). First, check if the two graphs have the same sets of eigenvalues. If they don't, then they are not isomorphic. If they do, and the eigenvalues have multiplicity one, then draw the pictures above. If the pictures are the same, up to horizontal or vertical flips, and no vertex is mapped to the same location as another, then by lining up the pictures we can recover the permutation.

As some vertices can map to the same location, this heuristic doesn't always work. We will learn about it to the extent to which it does. In particular, we will see that if every eigenvalue of two graphs G and H have multiplicity 1, then we can efficiently test whether or not they are isomorphic.

These algorithms have been extended to handle graph in which the multiplicity of every eigenvalue is bounded by a constant. But, there are graphs in which every non-trivial eigenvalue has large multiplicity. We will learn how to construct and analyze these, as they constitute fundamental examples and counter-examples to many natural conjectures. For example, here are the eigenvalues of a Latin Square Graph on 25 vertices. These are a type of Strongly Regular Graph.

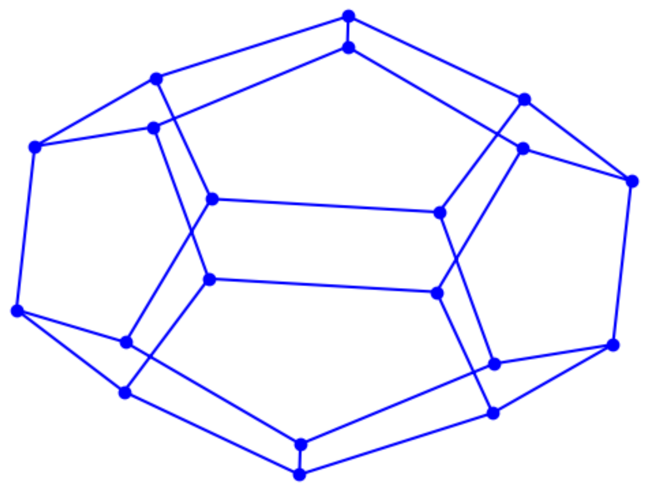
```
M = latin_square_graph(5);
println(eigvals(Matrix(lap(M))))
```

```
[0.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 15.0, 15.0, 1
```

All Latin Square Graphs of the same size have the same eigenvalues, whether or not they are isomorphic. We will learn some surprisingly fast (but still not polynomial time) algorithms for checking whether or not Strongly Regular Graphs are isomorphic.

1.7.4 Platonic Solids

Of course, somme graphs are not meant to be drawn in 3 dimensions. For example let's try this with the dodecahedron.

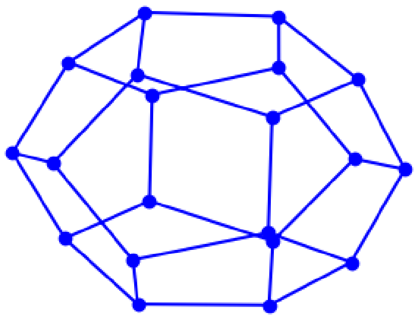


```
M = readIJV("dodec.txt")
spectral_drawing(M)
```

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.

```
E = eigen(Matrix(lap(M)))
println(E.values)
```

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.



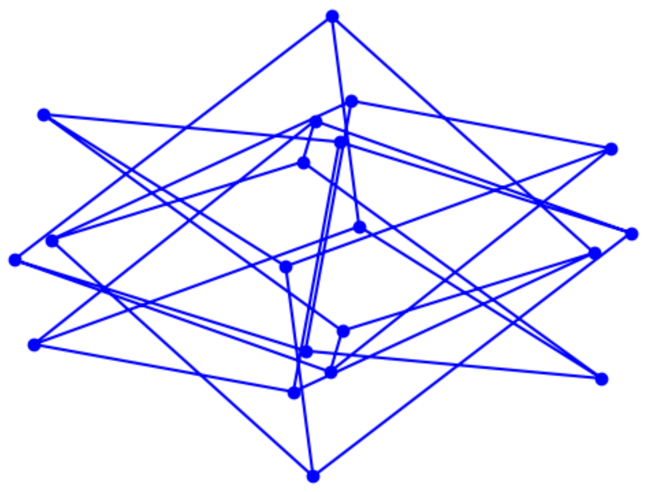
```
x = E.vectors[:,2]
y = E.vectors[:,3]
z = E.vectors[:,4]
pygui(true)
plot_graph(M, x, y, z; setaxis=false)
```

As you would guess, this happens for all Platonic solids. In fact, if you properly re-weight the edges, it happens for every graph that is the one-skeleton of a convex polytope. Let me state that more concretely. A weighted graph is a graph along with a weight function, w, mapping every edge to a positive number. The adjacency matrix of a weighted graph has the weights of edges as its entries, instead of 1s. The diagonal degree matrix of a weighted graph, D_G, has the weighted degrees on its diagonal. That is,

$$D_G(i,i) = \sum_{j:(i,j)\in E} w(i,j).$$

The Laplacian then becomes $L_G = D_G - A_G$. Given a convex polytope in $\mathbb{R}^d$, we can treat its 1-skeleton as a graph on its vertices. We will prove that there is a way of assigning weights to edges so that the second-smallest Laplacian eigenvalue has multiplicity d, and so that the corresponding eigenspace is spanned by the coordinate vectors of the vertices of the polytope.

Before we turn off the computer, let's take a look at the high-frequency eigenvectors of the dodecahedron.



```
x = E.vectors[:,11]
y = E.vectors[:,12]
z = E.vectors[:,10]
pygui(true)
plot_graph(M, x, y, z; setaxis=false)
```

1.7.5 The Fiedler Value

The second-smallest eigenvalue of the Laplacian matrix of a graph is zero if and only if the graph is disconnected. If G is disconnected, then we can partition it into two graphs G_1 and G_2 with no edges between them, and then write

$$\boldsymbol{L}_G = \begin{pmatrix} \boldsymbol{L}_{G_1} & 0 \\ 0 & \boldsymbol{L}_{G_2} \end{pmatrix}.$$

As the eigenvalues of $\boldsymbol{L}_G$ are the union, with multiplicity, of the eigenvalues of $\boldsymbol{L}_{G_1}$ and $\boldsymbol{L}_{G_2}$ we see that $\boldsymbol{L}_G$ inherits a zero eigenvalue from each. Conversely, if G is connected then we can show that the only vectors $\boldsymbol{x}$ for which $\boldsymbol{x}^T \boldsymbol{L}_G \boldsymbol{x} = 0$ are the constant vectors. If $\boldsymbol{x}$ is not constant and G is connected then there must be an edge (a, b) for which $\boldsymbol{x}(a) \neq \boldsymbol{x}(b)$. And, this edge will contribute a positive term to the sum (1.1).

Fiedler suggested that we make this qualitative observation quantitative and think of λ_2 as a measure of how well connected the graph is. For this reason, he called it the "Algebraic Connectivity" of a graph, and we call it the "Fiedler value".

Fiedler proved that the further λ_2 is from 0, the better connected the graph is. We will cover the ultimate extension of this result: Cheeger's inequality.

In short, we say that a graph is poorly connected if one can cut off many vertices by removing only a few edges. We measure how poorly connected it is by the ratio of these quantities (almost).

Cheeger's inequality gives a tight connection between this ratio and λ_2. If λ_2 is small, then for some t, the set of vertices

$$S_i \stackrel{\text{def}}{=} \{i : v_2(i) < t\}$$

may be removed by cutting much less than $|S_i|$ edges. This spectral graph partitioning heuristic has proved very successful in practice.

In general, it will be interesting to turn qualitative statements like this into quantitative ones. For example, we will see that the smallest eigenvalue of the diffusion matrix is zero if and only if the graph is bipartite. One can relate the magnitude of this eigenvalue to how far a graph is from being bipartite.

1.7.6 Bounding Eigenvalues

We will often be interested in the magnitudes of certain eigenvalues. For this reason, we will learn multiple techniques for proving bounds on eigenvalues. The most prominent of these will be proofs by test vectors and proofs by comparison with simpler graphs.

1.7.7 Planar Graphs

We will prove that graphs that can be drawn nicely must have small Fiedler value, and we will prove very tight results for planar graphs.

We will also see how to use the graph Laplacian to draw planar graphs: Tutte proved that if one reasonably fixes the locations of the vertices on a face of a planar graph and then lets the others settle into the positions obtained by treating the edges as springs, then one obtains a planar drawing of the graph!

1.7.8 Random Walks on Graphs

Spectral graph theory is one of the main tools we use for analyzing random walks on graphs. We will spend a few lecture on this theory, connect it to Cheeger's inequality, and use tools developed to study random walks to derive a fascinating proof of Cheeger's inequality.

1.7.9 Expanders

We will be particularly interested in graphs that are very well connected. These are called *expanders*. Roughly speaking, expanders are sparse graphs (say a number of edges linear in the number of vertices), in which λ_2 is bounded away from zero by a constant. They are among the most important examples of graphs, and play a prominent role in Theoretical Computer Science.

Expander graphs have numerous applications. We will see how to use random walks on expander graphs to construct pseudo-random generators *about which one can actually prove something.* We will also use them to construct good error-correcting codes.

Error-correcting codes and expander graphs are both fundamental objects of study in the field of Extremal Combinatorics and are extremely useful. If students in the class have not learned about these, I will teach about them. We will also use error-correcting codes to construct crude expander graphs.

We will learn at least one construction of good expanders. The best expanders are the Ramanujan graphs. These were first constructed by Margulis and Lubotzky, Phillips and Sarnak. We might finish the class by proving the existence of Ramanujan graphs.

1.7.10 Approximations of Graphs

We will ask what it means for one graph to approximate another. Given graphs G and H, we will measure how well G approximates H by the closeness of their Laplacian quadratic forms. We will see that expanders are precisely the sparse graphs that provide good approximations of the complete graph, and we will use this perspective for most of our analysis of expanders. We will show that every graph can be well-approximated by a sparse graph through a process called *sparsification.*

1.7.11 Solving equations in and computing eigenvalues of Laplacians

We will also ask how well a graph can be approximated by a tree, and see that low-stretch spanning-trees provide good approximations under this measure.

My motivation for this material is not purely graph-theoretic. Rather, it is inspired by the need to design fast algorithms for computing eigenvectors of Laplacian matrices and for solving linear equations in Laplacian matrices. This later problem arises in numerous contexts, including the solution of elliptic PDEs by the finite element method, the solution of network flow problems by interior point algorithms, and in classification problems in Machine Learning.

In fact, our definition of graph approximation is designed to suit the needs of the Preconditioned Conjugate Gradient algorithm. We may finish the semester by learning how these algorithms work.

1.8 Eigenvalues and Optimization

One of the reasons that the eigenvalues of matrices have meaning is that they arise as the solution to natural optimization problems. We will spend a lot of time on this connection next lecture. For now, we start with one result in this direction. Observe that its proof does not require the spectral theorem.

Theorem 1.8.1. *Let $\boldsymbol{M}$ be a symmetric matrix and let $\boldsymbol{x}$ be a non-zero vector that maximizes the Rayleigh quotient with respect to $\boldsymbol{M}$:*

$$\frac{\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}.$$

Then, $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{M}$ with eigenvalue equal to the Rayleigh quotient. Moreover, this eigenvalue is the largest eigenvalue of $\boldsymbol{M}$.

Proof. We first observe that the maximum is achieved: As the Rayleigh quotient is homogeneous, it suffices to consider unit vectors $\boldsymbol{x}$. As the set of unit vectors is a closed and compact set, the maximum is achieved on this set.

Now, let $\boldsymbol{x}$ be a non-zero vector that maximizes the Rayleigh quotient. We recall that the gradient of a function at its maximum must be the zero vector. Let's compute that gradient.

We have

$$\nabla \boldsymbol{x}^T \boldsymbol{x} = 2\boldsymbol{x},$$

and

$$\nabla \boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x} = 2\boldsymbol{M}\boldsymbol{x}.$$

So,

$$\nabla \frac{\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} = \frac{(\boldsymbol{x}^T \boldsymbol{x})(2\boldsymbol{M}\boldsymbol{x}) - (\boldsymbol{x}^T \boldsymbol{M}\boldsymbol{x})(2\boldsymbol{x})}{(\boldsymbol{x}^T \boldsymbol{x})^2}.$$

In order for this to be zero, we must have

$$\boldsymbol{M}\boldsymbol{x} = \frac{\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} \boldsymbol{x}.$$

That is, if and only if $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{M}$ with eigenvalue equal to its Rayleigh quotient. $\square$

1.9 Exercises

The following exercises are for your own practice. They are intended as a review of fundamental linear algebra. I have put the solutions in a separate file that you can find on Classes V2. I recommend that you try to solve all of these before you look at the solutions, so that you can get back in practice at doing linear algebra.

1. Orthogonal eigenvectors. Let $\boldsymbol{M}$ be a symmetric matrix, and let $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ be vectors so that

$$\boldsymbol{M}\boldsymbol{\psi} = \mu\boldsymbol{\psi} \quad \text{and} \quad \boldsymbol{M}\boldsymbol{\phi} = \nu\boldsymbol{\phi}.$$

Prove that if $\mu \neq \nu$ then $\boldsymbol{\psi}$ must be orthogonal to $\boldsymbol{\phi}$. Note that your proof should exploit the symmetry of $\boldsymbol{M}$, as this statement is false otherwise.

2. Invariance under permutations.

Let $\boldsymbol{\Pi}$ be a permutation matrix. That is, there is a permutation $\pi : V \to V$ so that

$$\boldsymbol{\Pi}(u, v) = \begin{cases} 1 & \text{if } u = \pi(v), \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Prove that if

$$\boldsymbol{M}\boldsymbol{\psi} = \lambda\boldsymbol{\psi},$$

then

$$\left(\boldsymbol{\Pi}\boldsymbol{M}\boldsymbol{\Pi}^T\right)(\boldsymbol{\Pi}\boldsymbol{\psi}) = \lambda(\boldsymbol{\Pi}\boldsymbol{\psi}).$$

That is, permuting the coordinates of the matrix merely permutes the coordinates of the eigenvectors, and does not change the eigenvalues.

3. Invariance under rotations.

Let $\boldsymbol{Q}$ be an orthonormal matrix. That is, a matrix such that $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$. Prove that if

$$\boldsymbol{M}\boldsymbol{\psi} = \lambda\boldsymbol{\psi},$$

then

$$\left(\boldsymbol{Q}\boldsymbol{M}\boldsymbol{Q}^T\right)(\boldsymbol{Q}\boldsymbol{\psi}) = \lambda(\boldsymbol{Q}\boldsymbol{\psi}).$$

4. Similar Matrices.

A matrix $\boldsymbol{M}$ is similar to a matrix $\boldsymbol{B}$ if there is a non-singular matrix $\boldsymbol{X}$ such that $\boldsymbol{X}^{-1}\boldsymbol{M}\boldsymbol{X} = \boldsymbol{B}$. Prove that similar matrices have the same eigenvalues.

5. Spectral decomposition.

Let $\boldsymbol{M}$ be a symmetric matrix with eigenvalues $\lambda_1, \dots, \lambda_n$ and let $\boldsymbol{\psi}_1, \dots, \boldsymbol{\psi}_n$ be a corresponding set of orthonormal column eigenvectors. Let $\boldsymbol{\Psi}$ be the orthonormal matrix whose ith column is $\boldsymbol{\psi}_i$. Prove that

$$\boldsymbol{\Psi}^T\boldsymbol{M}\boldsymbol{\Psi} = \boldsymbol{\Lambda},$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix with $\lambda_1, \dots, \lambda_n$ on its diagonal. Conclude that

$$\boldsymbol{M} = \boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^T = \sum_{i \in V} \lambda_i \boldsymbol{\psi}_i \boldsymbol{\psi}_i^T.$$