Table of Contents

This is a Jupyter Notebook that contains Julia code I will run in the first lecture of Spectral Graph Theory. I find experiments to be incredibly useful when working on spectral graph theory. They help me figure out what is true, and they help me find counterexamples to my conjectures.

If you want to try using this, you will need to install Jupyter (via Python), Julia and IJulia. You also need my package, called `Laplacians.jl`. It may be added in Julia via

```
Using Pkg
Pkg.add("Laplacians")
```

In [56]:

```
using Laplacians
using LinearAlgebra
using Plots
using SparseArrays
using FileIO
using JLD2
using Random
```

In [57]:

```
gr()
```

Out[57]:

```
Plots.GRBackend()
```

Path Graphs

In [58]:

```
M = path_graph(4)
```

Out[58]:

```
4×4 SparseMatrixCSC{Float64,Int64} with 6 stored entries:
  [2, 1]  =  1.0
  [1, 2]  =  1.0
  [3, 2]  =  1.0
  [2, 3]  =  1.0
  [4, 3]  =  1.0
  [3, 4]  =  1.0
```

In [59]:

```
Matrix(M)
```

Out[59]:

```
4×4 Array{Float64,2}:
 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
```

In [60]:

```
Matrix(lap(M))
```

Out[60]:

```
4×4 Array{Float64,2}:
  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
```

In [61]:

```
L = lap(path_graph(10))
;
```

In [62]:

```
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]
```

In [63]:

```
E.vectors[:,1]
```

Out[63]:

```
10-element Array{Float64,1}:
 0.31622776601683755
 0.31622776601683716
 0.31622776601683766
 0.3162277660168381
 0.31622776601683855
 0.3162277660168381
 0.3162277660168385
 0.31622776601683805
 0.3162277660168378
 0.3162277660168378
```

In [64]:

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

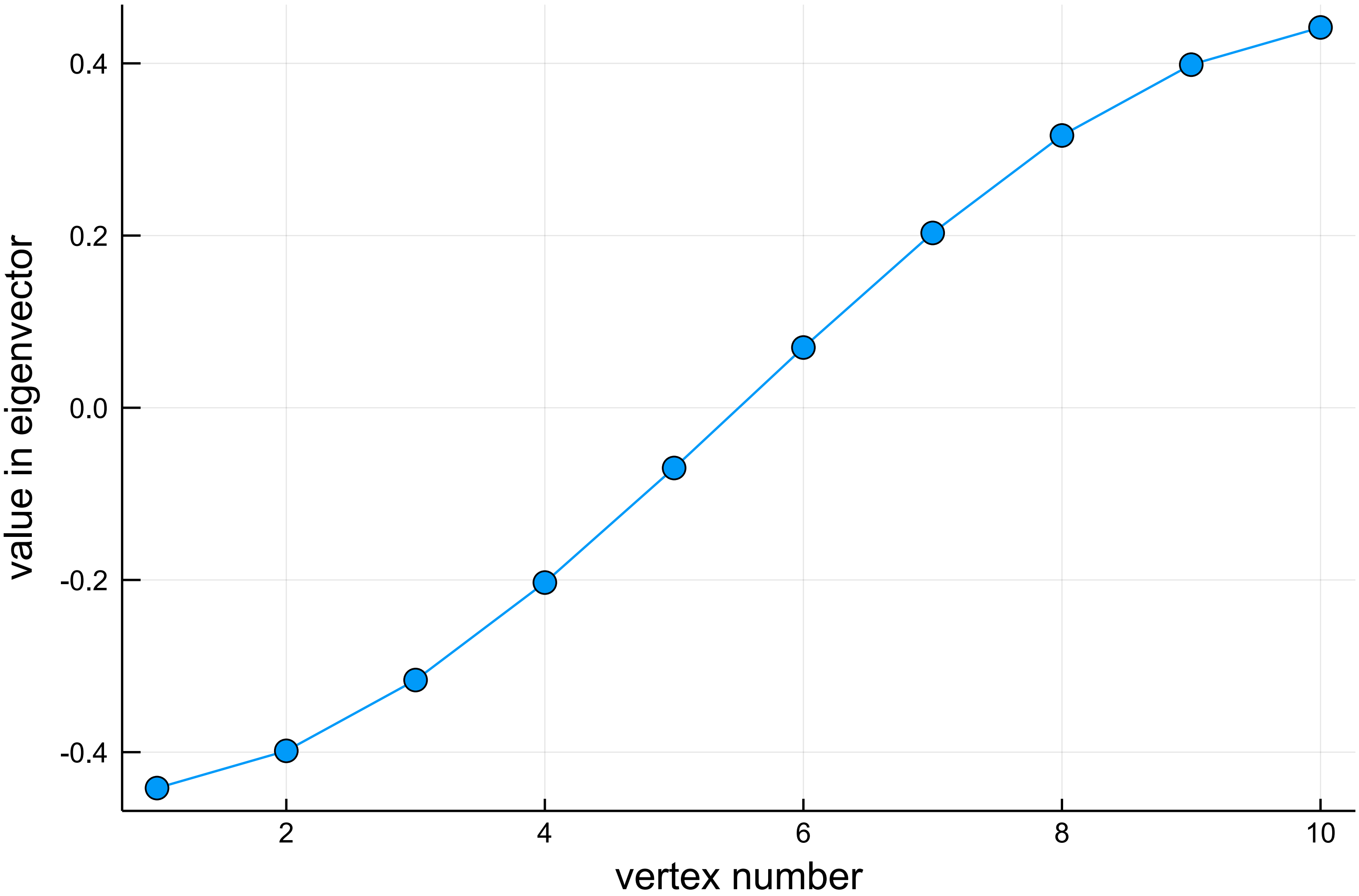
Out[64]:

```
10-element Array{Float64,1}:
 -0.44170765403093937
 -0.39847023129620024
 -0.316227766016838
 -0.20303072371134553
 -0.06995961957075425
  0.06995961957075386
  0.203030723711345 7
  0.31622776601683766
  0.3984702312961997
  0.4417076540309382
```

In [65]:

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

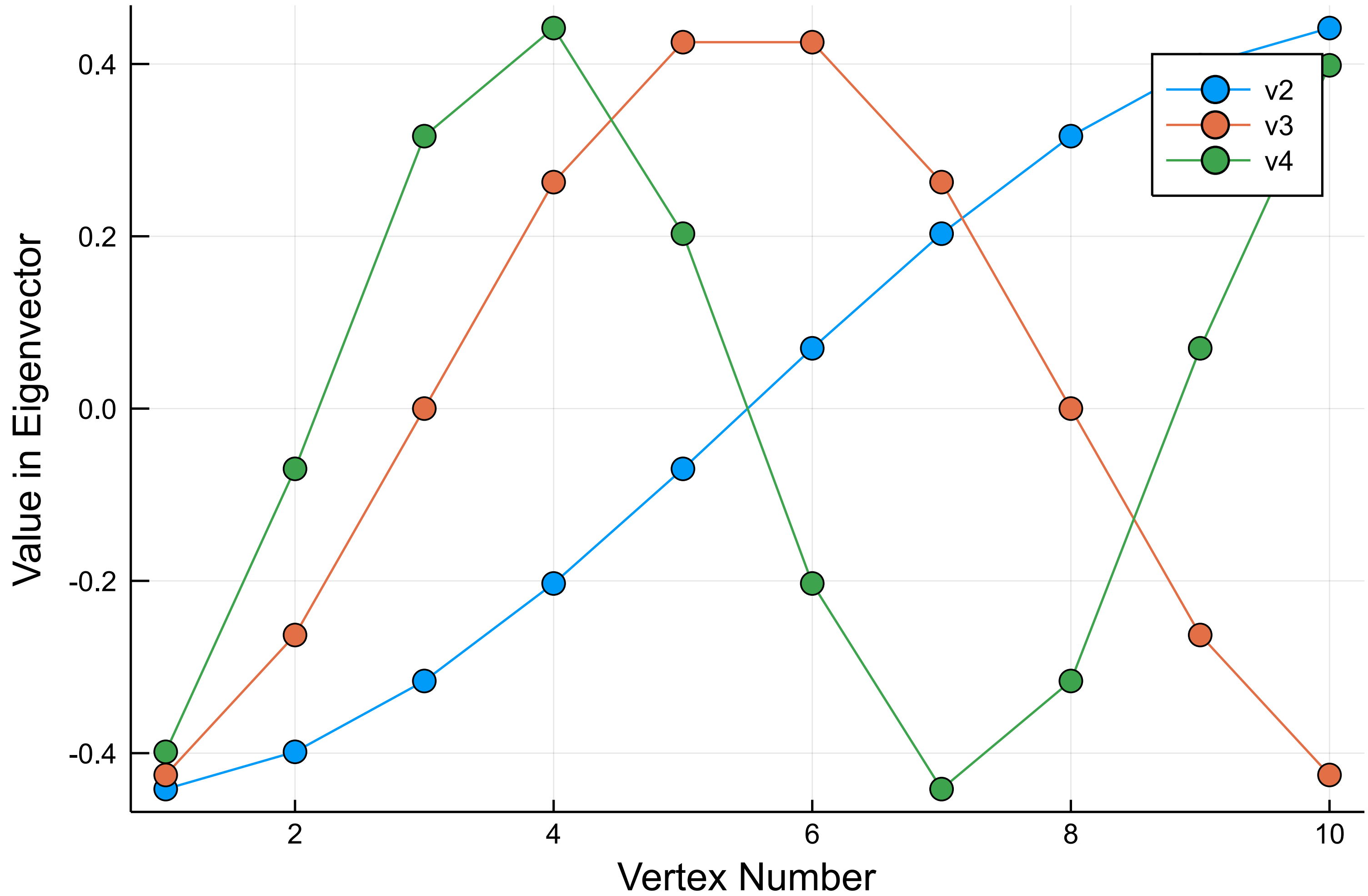
Out[65]:



In [66]:

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

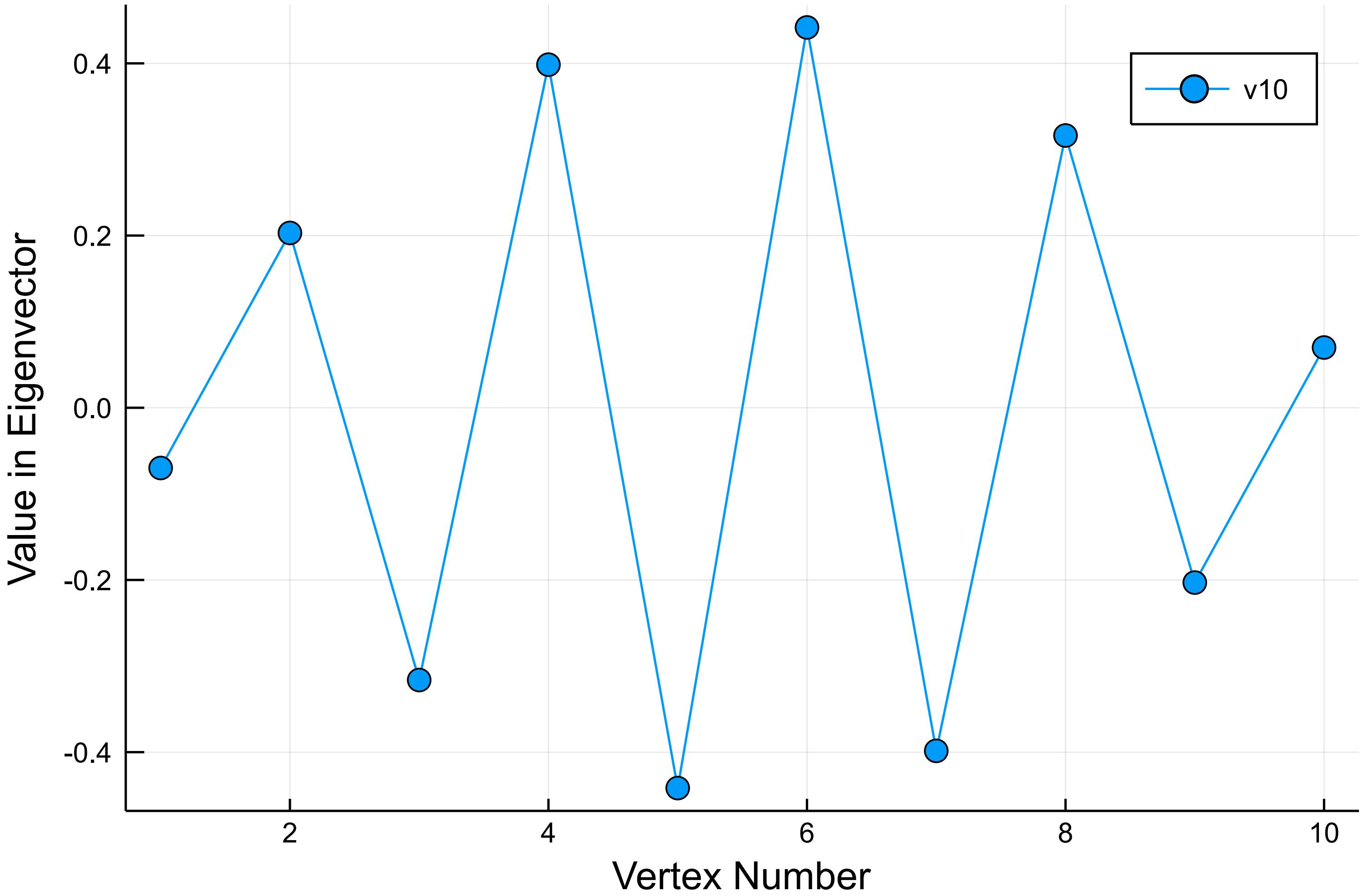
Out[66]:



In [67]:

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

Out[67]:



Spectral Graph Drawing -- a grid graph

In [68]:

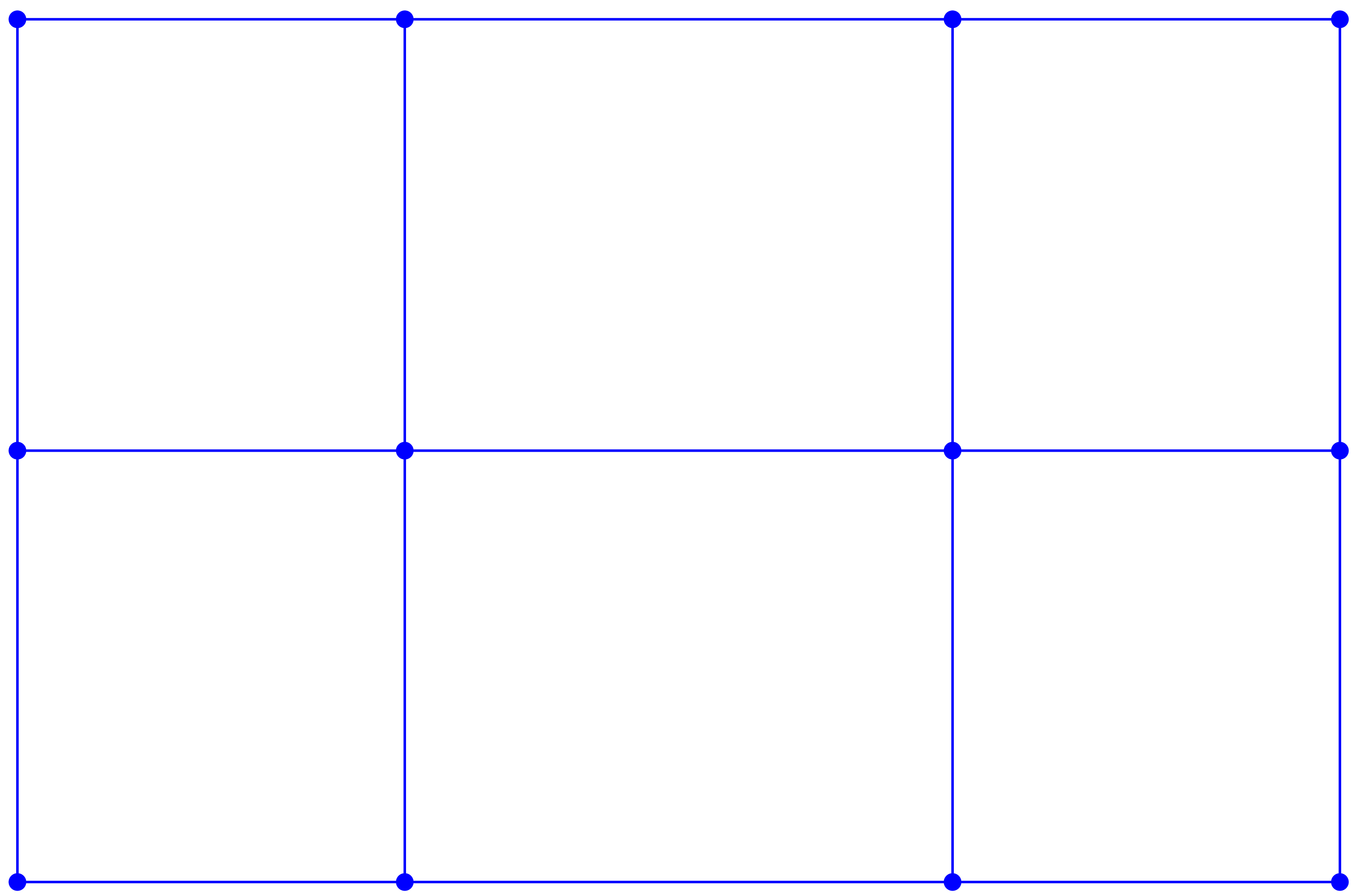
```
M = grid2(3,4)
L = lap(M)
E = eigen(Matrix(L))
V = E.vectors[:,2:3]
```

Out[68]:

```
12×2 Array{Float64,2}:
 -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 [69]:

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



Spectral Graph Drawing -- The Yale Logo

In [70]:

```
@load "YALE.jld2"
```

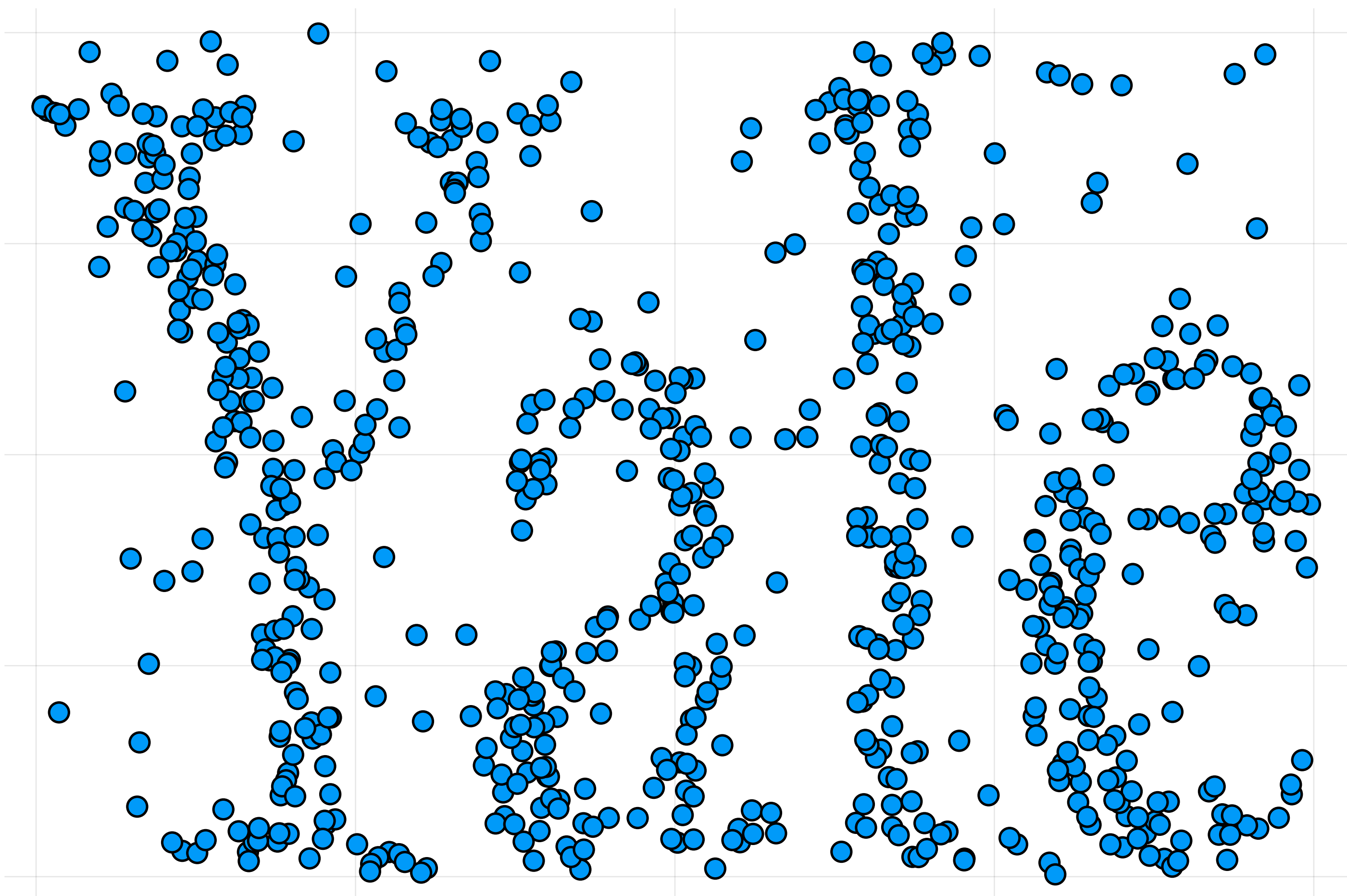
Out[70]:

```
4-element Array{Symbol,1}:
 :a
 :xy
 :v2
 :v3
```

In [71]:

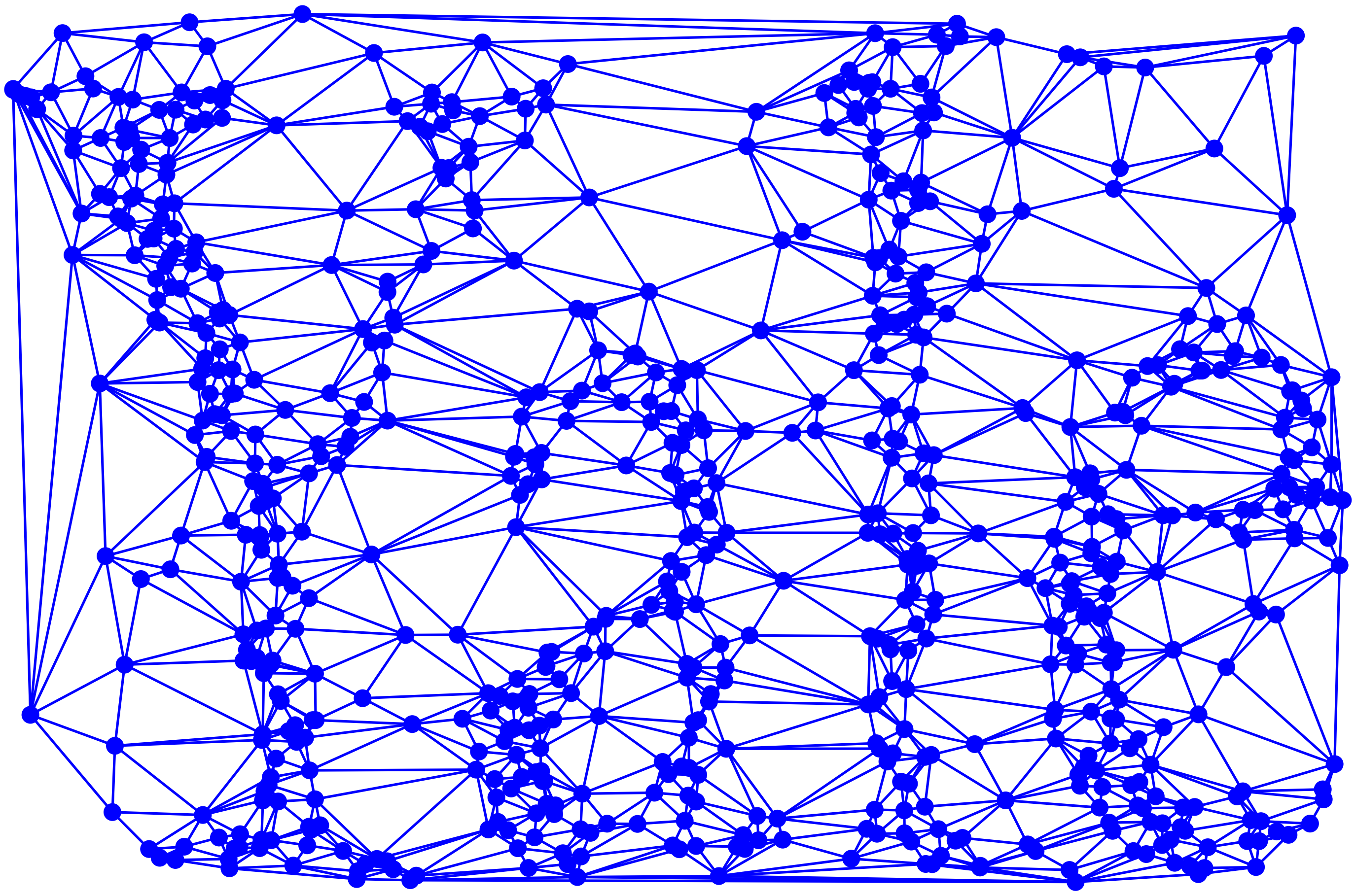
```
ax = scatter(xy[:,1],xy[:,2],legend=false, axis=false)
```

Out[71]:



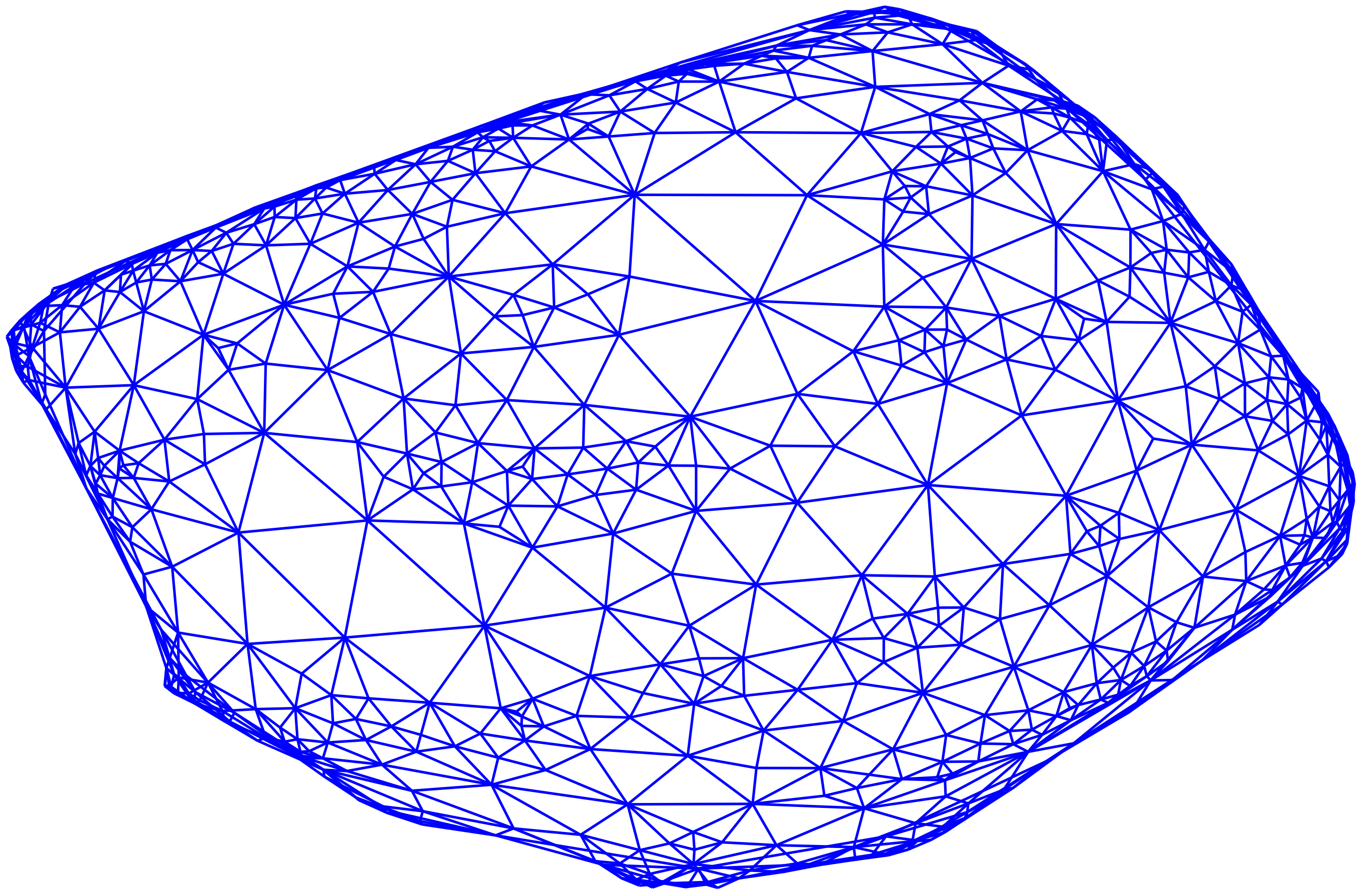
In [72]:

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



In [73]:

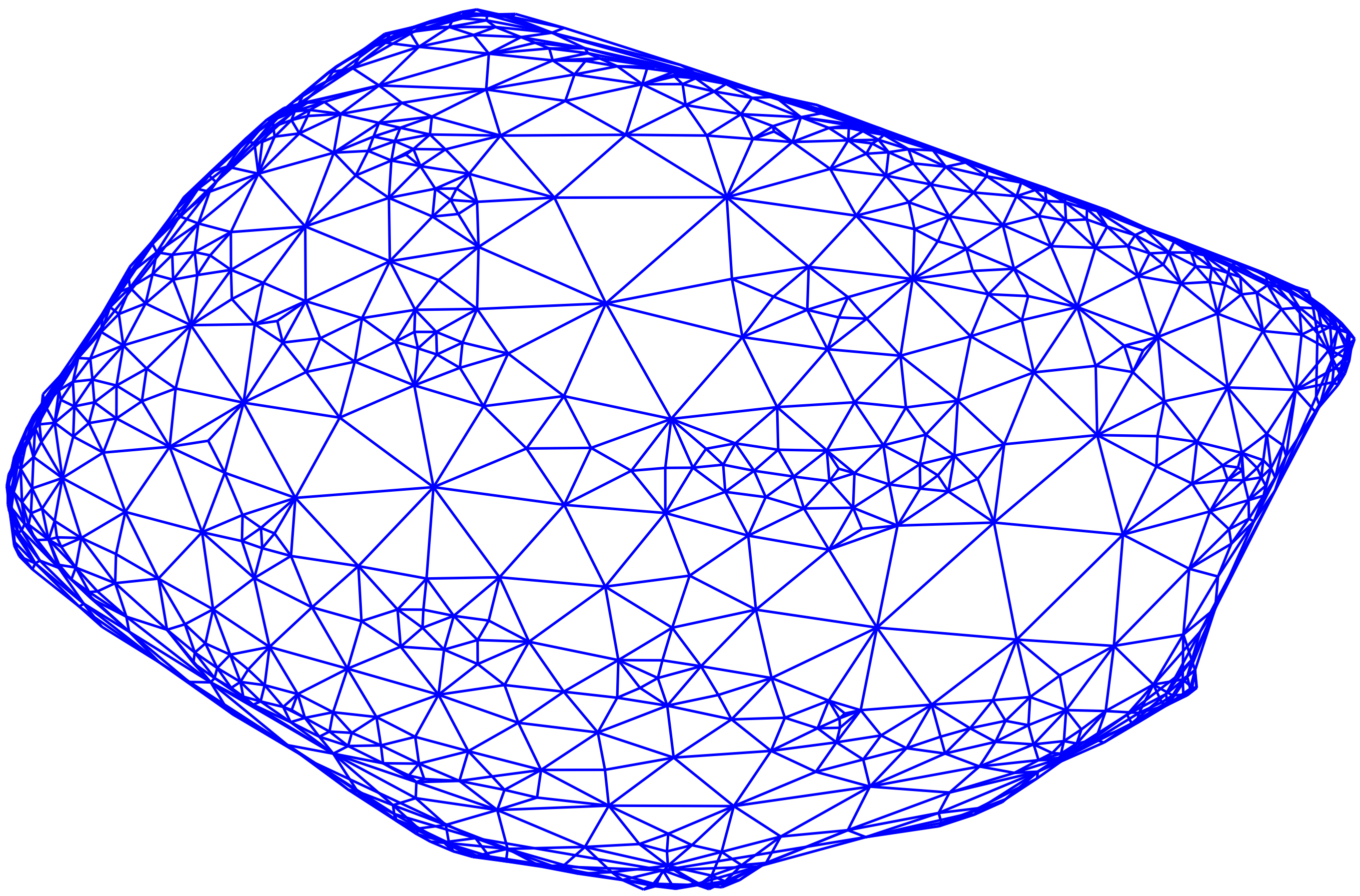
```
plot_graph(a, v2,v3, dots=false);
```



Isomorphism

In [74]:

```
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);
```



The dodecahedron

In [75]:

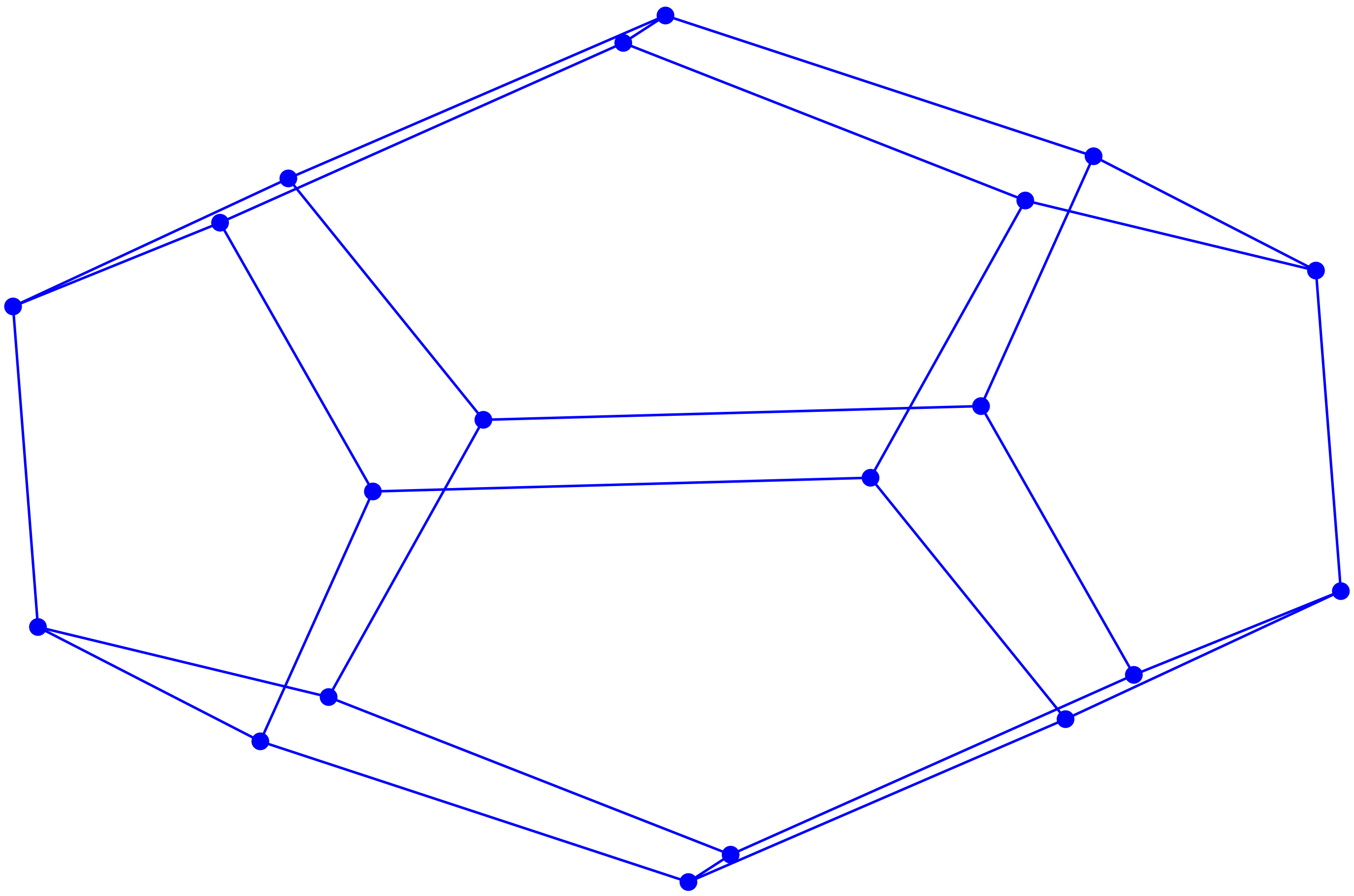
```
M = read_graph("dodec.txt")
```

Out[75]:

```
20×20 SparseMatrixCSC{Float64,Int64} with 60 stored entries:
  [2 ,  1]  =  1.0
  [5 ,  1]  =  1.0
  [16,  1]  =  1.0
  [1 ,  2]  =  1.0
  [3 ,  2]  =  1.0
  [15,  2]  =  1.0
  [2 ,  3]  =  1.0
  [4 ,  3]  =  1.0
  [13,  3]  =  1.0
  [3 ,  4]  =  1.0
  [5 ,  4]  =  1.0
  [8 ,  4]  =  1.0
  ⋮
  [15, 17]  =  1.0
  [16, 17]  =  1.0
  [20, 17]  =  1.0
  [6 , 18]  =  1.0
  [16, 18]  =  1.0
  [19, 18]  =  1.0
  [9 , 19]  =  1.0
  [18, 19]  =  1.0
  [20, 19]  =  1.0
  [12, 20]  =  1.0
  [17, 20]  =  1.0
  [19, 20]  =  1.0
```

In [76]:

```
spectral_drawing(M);
```



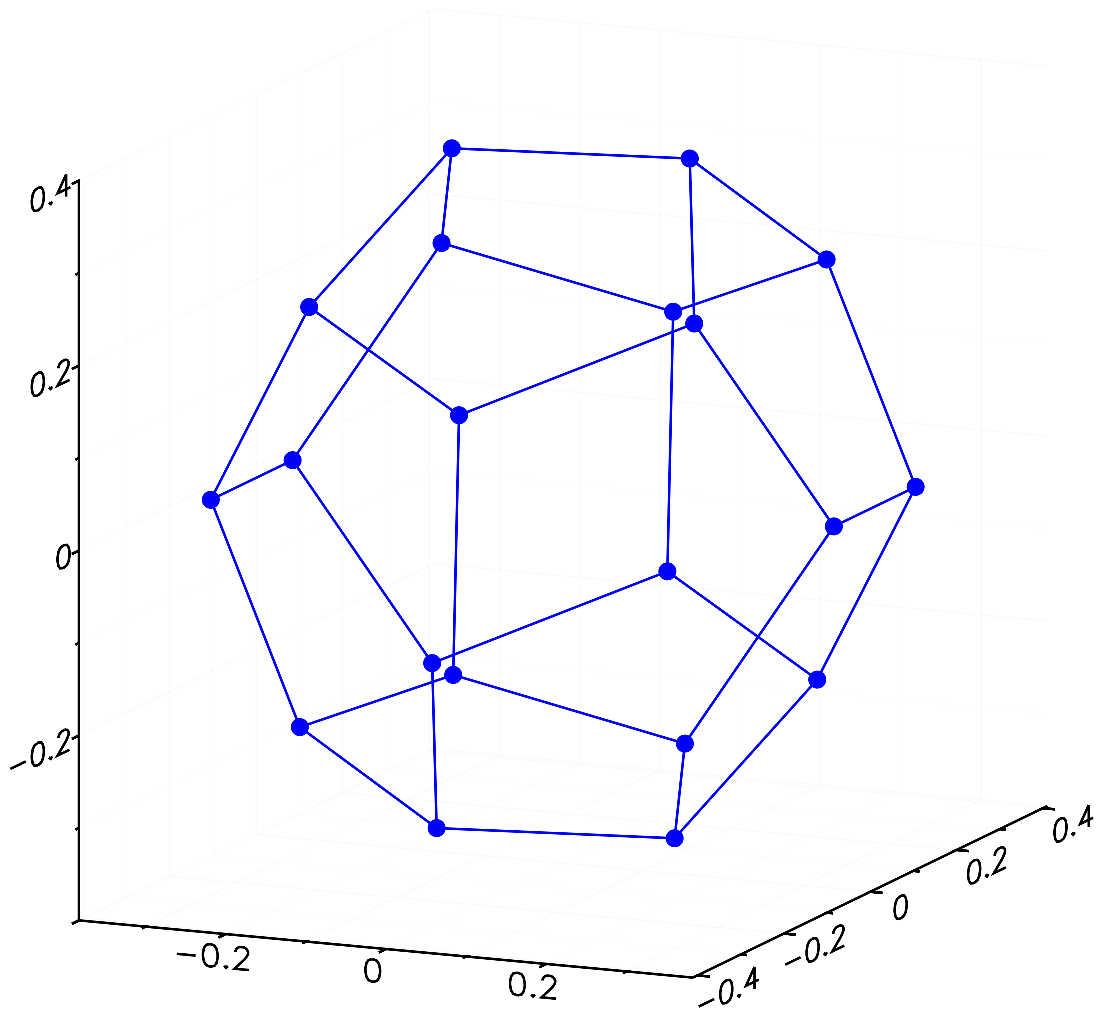
In [77]:

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

```
[-8.88178e-16, 0.763932, 0.763932, 0.763932, 2.0, 2.0, 2.0, 2.0, 2.0
, 3.0, 3.0, 3.0, 3.0, 5.0, 5.0, 5.0, 5.0, 5.23607, 5.23607, 5.23607]
```

In [78]:

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



In [79]:

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

In [80]:

```
x = E.vectors[:,20]
y = E.vectors[:,19]
plot_graph(M, x, y; setaxis=false);
```

In [81]:

```
x = E.vectors[:,20]
y = E.vectors[:,19]
z = E.vectors[:,18]
plot_graph(M, x, y, z; setaxis=false);
```

In [82]:

```
E.values
```

Out[82]:

```
20-element Array{Float64,1}:
 -8.881784197001252e-16
  0.7639320225002125
  0.7639320225002131
  0.7639320225002177
  1.999999999999998
  2.0000000000000018
  2.0000000000000027
  2.000000000000003
  2.000000000000004
  3.0
  3.0000000000000004
  3.0000000000000013
  3.000000000000002
  4.999999999999998
  4.999999999999999
  5.0
  5.000000000000001
  5.236067977499789
  5.23606797749979
  5.23606797749979
```