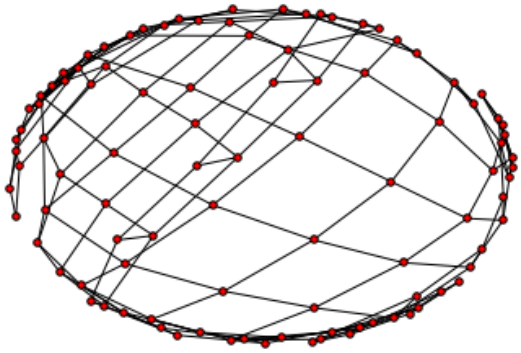
In [1]:
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import numpy.linalg as lin
%matplotlib inline
```

In [2]:
```
# a simple function to compute a dictionary with locations of nodes, as given
 by vectors x and y
def embed_nodes(g, x, y):
    return dict((g.nodes()[i], (x[i],y[i])) for i in range(len(g.nodes())))
```

In [3]:
```
#Create a grid graph
gr = nx.grid_2d_graph(10, 10)
```

In [64]:
```
#Let's try to draw it:
nx.draw(gr, node_size=20)
```

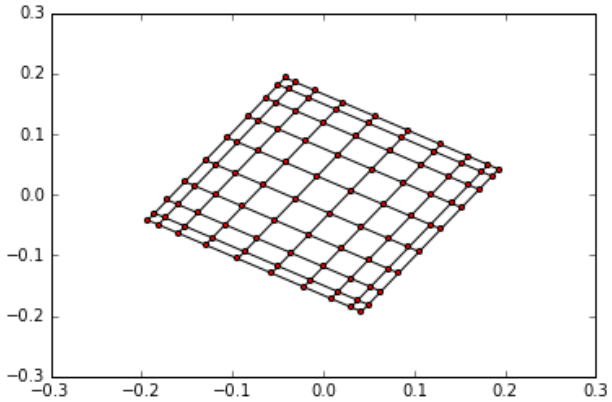


In [65]:
```
lapgr = nx.laplacian_matrix(gr)
```

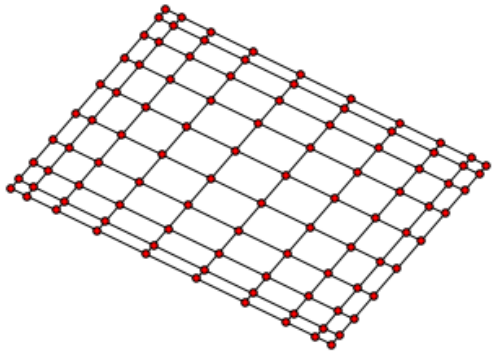
In [66]:
```
#compute eigens:
(w,v) = lin.eigh(lapgr.todense())
```

In [67]:
```
# compute embedding (positions) using the eigen vectors corresponding to eige
n values 1 and 2
# (not eig zero)
emb = embed_nodes(gr,v[:,1].ravel().tolist()[0],v[:,2].ravel().tolist()[0])
```

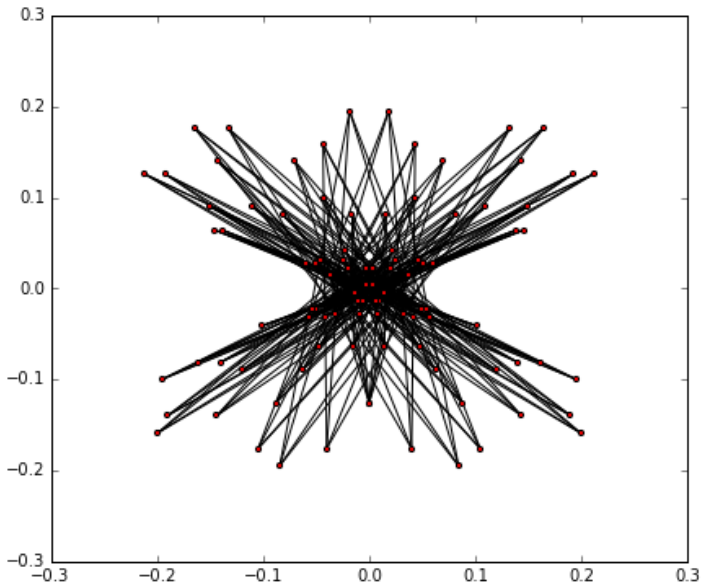
In [68]:
```
#plt.figure(figsize=(8,8))
nx.draw_networkx(gr, pos=emb, node_size=10, with_labels=False)
```



In [69]:
```
nx.draw_spectral(gr, node_size=20)
```

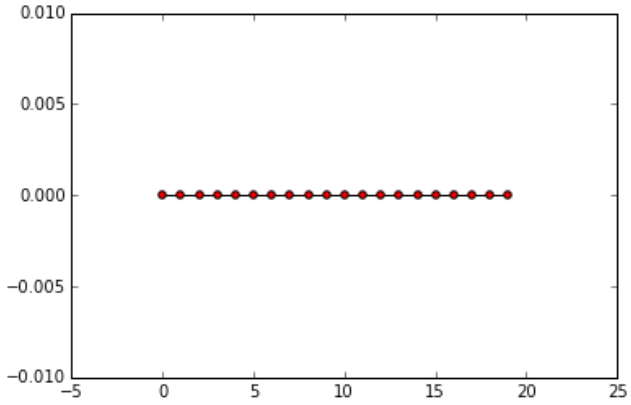


In [73]:
```
#draw with the highest eigen values
plt.figure(figsize=(7,6))
emb = embed_nodes(gr,v[:,98].ravel().tolist()[0],v[:,99].ravel().tolist()[0])
nx.draw_networkx(gr, pos=emb, node_size=10, with_labels=False)
```



In [74]:
```
#make a 1-d grid
grid1d = nx.grid_2d_graph(20,1)
#nx.draw(grid1d, with_labels=True)
```

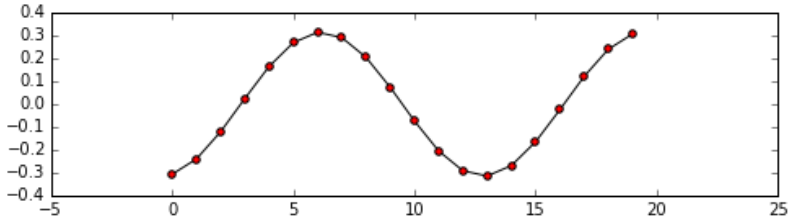
In [75]:
```
#manually draw them on a line
emb = dict((grid1d.nodes()[i], (grid1d.nodes()[i][0],grid1d.nodes()[i][1])) f
or i in range(len(grid1d.nodes())))
nx.draw_networkx(grid1d, pos=emb, node_size=20, with_labels=False)
```



In [76]:
```
#compute laplacian and eigen vectors
lap1gr = nx.laplacian_matrix(grid1d)
(w,v) = lin.eigh(lap1gr.todense())
w
```

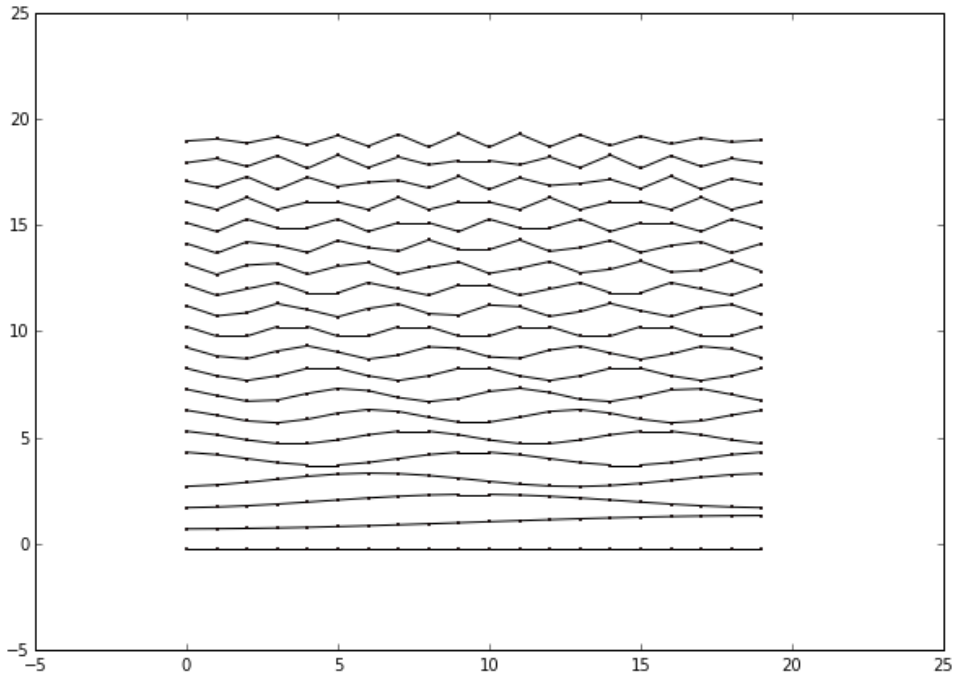
Out[76]:
```
array([  3.76788031e-16,   2.46233188e-02,   9.78869674e-02,
         2.17986952e-01,   3.81966011e-01,   5.85786438e-01,
         8.24429495e-01,   1.09201900e+00,   1.38196601e+00,
         1.68713107e+00,   2.00000000e+00,   2.31286893e+00,
         2.61803399e+00,   2.90798100e+00,   3.17557050e+00,
         3.41421356e+00,   3.61803399e+00,   3.78201305e+00,
         3.90211303e+00,   3.97537668e+00])
```

In [78]:
```
# draw the jth eigen vector
plt.figure(figsize=(8,2))
j=3
emb = dict((grid1d.nodes()[i], (grid1d.nodes()[i][0], v[:,j].ravel().tolist()
[0][i])) for i in range(len(grid1d.nodes())))
nx.draw_networkx(grid1d, pos=emb, node_size=20, with_labels=False)
```

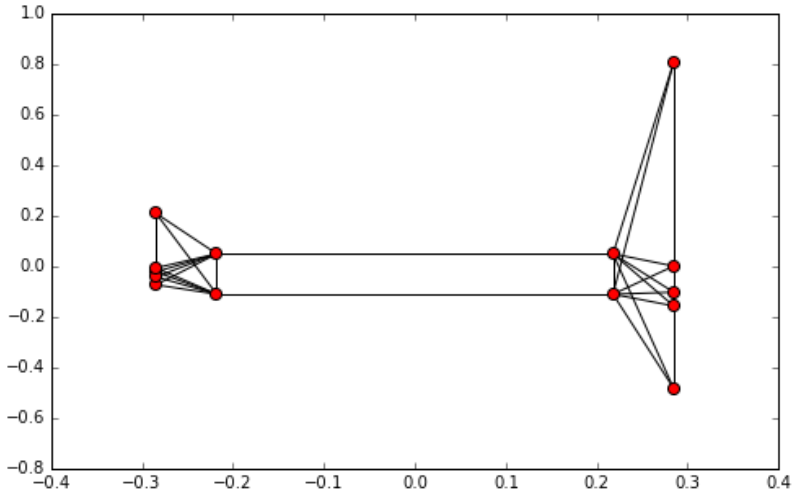


In [79]:
```
#let's try to plot all of them:
plt.figure(figsize=(10,7))

for j in range(20):
    emb = dict((grid1d.nodes()[i], (grid1d.nodes()[i][0], j+v[:,j].ravel().to
list()[0][i])) for i in range(len(grid1d.nodes())))
    nx.draw_networkx(grid1d, pos=emb, node_size=1, with_labels=False)
```

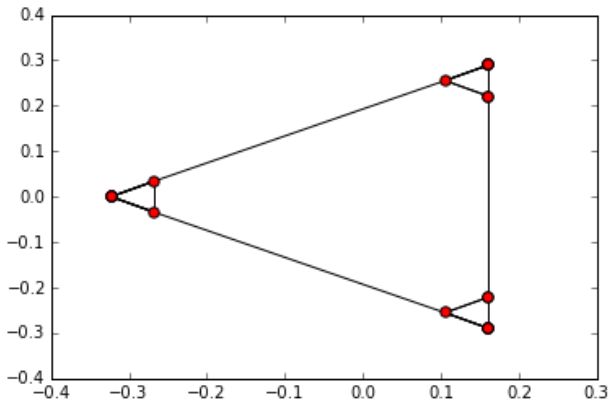


In [80]:
```
#community detection:
bg = nx.barbell_graph(7,0)
bg.add_edge(9,1)
lpbg = nx.laplacian_matrix(bg)
(w,v) = lin.eigh(lpbg.todense())
emb = embed_nodes(bg,v[:,1].ravel().tolist()[0],v[:,2].ravel().tolist()[0])
plt.figure(figsize=(8,5))
nx.draw_networkx(bg, pos=emb, node_size=50, with_labels=False)
```



In [81]:
```
#let's add one more community:

c = nx.complete_graph(7)
h = nx.disjoint_union(bg,c)
h.add_edge(20,10)
h.add_edge(17,2)
lph = nx.laplacian_matrix(h)
(w,v) = lin.eigh(lph.todense())
emb = embed_nodes(h,v[:,1].ravel().tolist()[0],v[:,2].ravel().tolist()[0])
nx.draw_networkx(h, pos=emb, node_size=40, with_labels=False)
```



In [ ]: