Coloring Communities

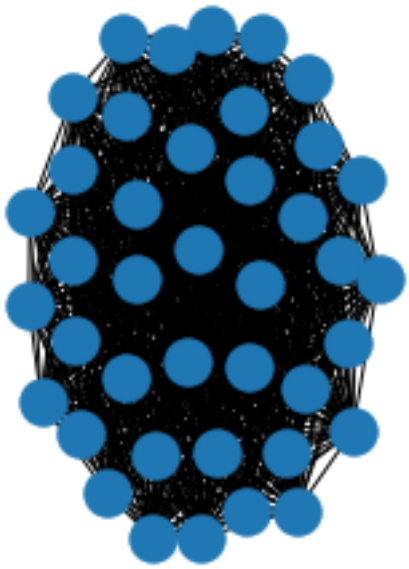
April 30, 2020

[79]:
```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from random import random, shuffle
import itertools
from L_svd import L_svd
from numpy.fft import fft, ifft
from scipy.linalg import svd
from mpl_toolkits.mplot3d import Axes3D
```

1 Basic Setup

[57]:
```
# make a fully connected graph
size = 40
nodes = [x for x in range(0, size)]
s = (size, size)
mCom = np.ones(s)
g = nx.Graph(mCom)
```

[58]:
```
# look at it for fun
plt.subplot(121)
nx.draw(g)
```



[59]:
```
# separate into 2 communities
shuffle(nodes)

first_community = nodes[:int(size/2)]
second_community = list(set(nodes) - set(first_community))

first_community, second_community
```

[59]:
```
([30, 34, 39, 21, 11, 10, 5, 20, 0, 28, 27, 16, 22, 14, 23, 35, 8, 29, 13, 12],
 [1, 2, 3, 4, 6, 7, 9, 15, 17, 18, 19, 24, 25, 26, 31, 32, 33, 36, 37, 38])
```

[60]:
```
# going to use some python slickness here
connections = [i for i in itertools.combinations_with_replacement(nodes, 2)]
```

[61]:
```
first_community_connections = [i for i in itertools.
 ↪permutations(first_community, 2)]
second_community_connections = [i for i in itertools.
 ↪permutations(second_community, 2)]
loops = [(x,x) for x in range(0,size)]

connections = list(set(connections) - set(first_community_connections))
connections = list(set(connections) - set(loops))
illegal_connections = list(set(connections) - set(second_community_connections))
```

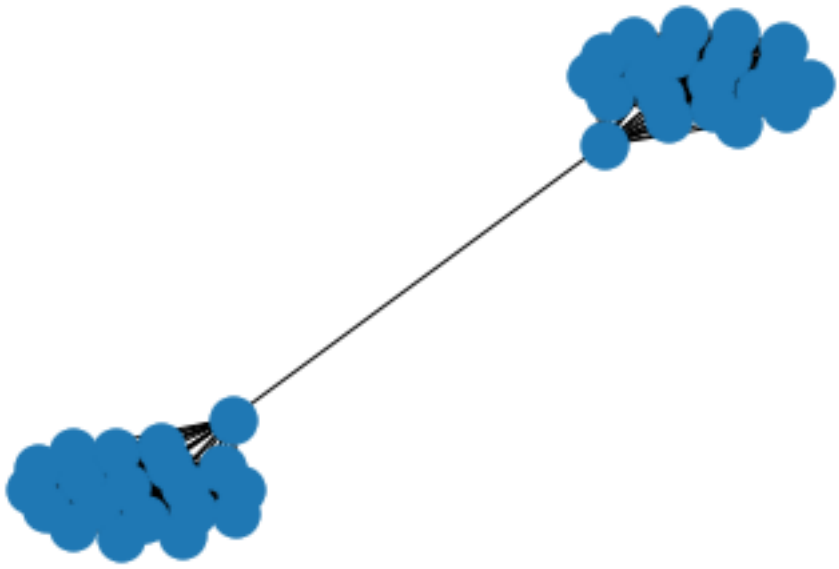
[62]:
```
shuffle(illegal_connections)
link = illegal_connections.pop()
illegal_connections = list(set(illegal_connections) - set(link))
```

```
testMcom = mCom.copy()
link, testMcom.shape
```

[62]: ((20, 3), (40, 40))

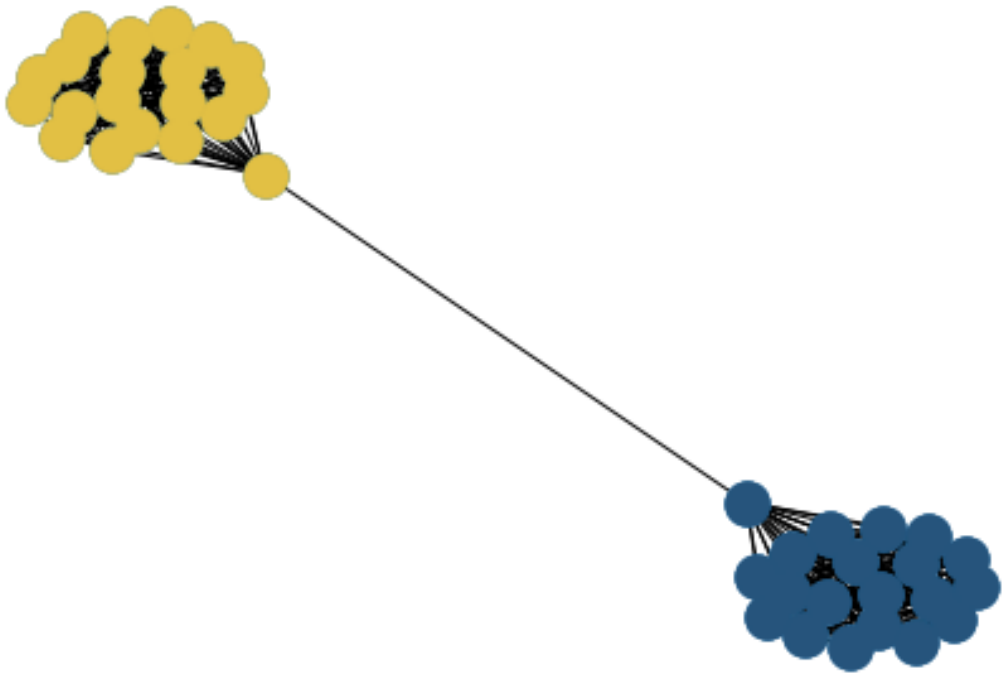
[63]:
```
results = np.zeros((size, len(illegal_connections), size))

for i, x in enumerate(illegal_connections, start=0):
    testMcom[x[0]][x[1]] = 0
    testMcom[x[1]][x[0]] = 0

    #print(results.shape)
    #print(testMcom.shape)

    results[:,i,:] = testMcom.copy()
```

[64]:
```
g_draw = nx.Graph(testMcom, with_labels=True, font_weight='bold')
plt.figure()
plt.plot()
nx.draw(g_draw)
plt.show()
```



[65]:
```
G3 = nx.Graph(testMcom, with_labels=True, font_weight='bold')

pos = nx.spring_layout(G3)
```

```
# Draw the graph, but don't color the nodes
nx.draw(G3, pos)

#For each community list, draw the nodes, giving it a specific color.
nx.draw_networkx_nodes(G3, pos, nodelist=first_community, node_color='#26547C')
nx.draw_networkx_nodes(G3, pos, nodelist=second_community, node_color='#E2C044')
plt.show()
```



[66]:
```
t = np.array(results)
t.shape
```

[66]: (40, 399, 40)

2 Tensor Coloring

[16]:
```
# file with all converted tensor operations
from tensorOps import *
```

[67]:
```
U, S, V = t_svd(t)
```

```
(40, 399, 40)
```

[72]:
```
Mt = t_prod(t_tran(U[:,0:3,:]), t)

Mt = np.real(Mt)

Mt.shape
```

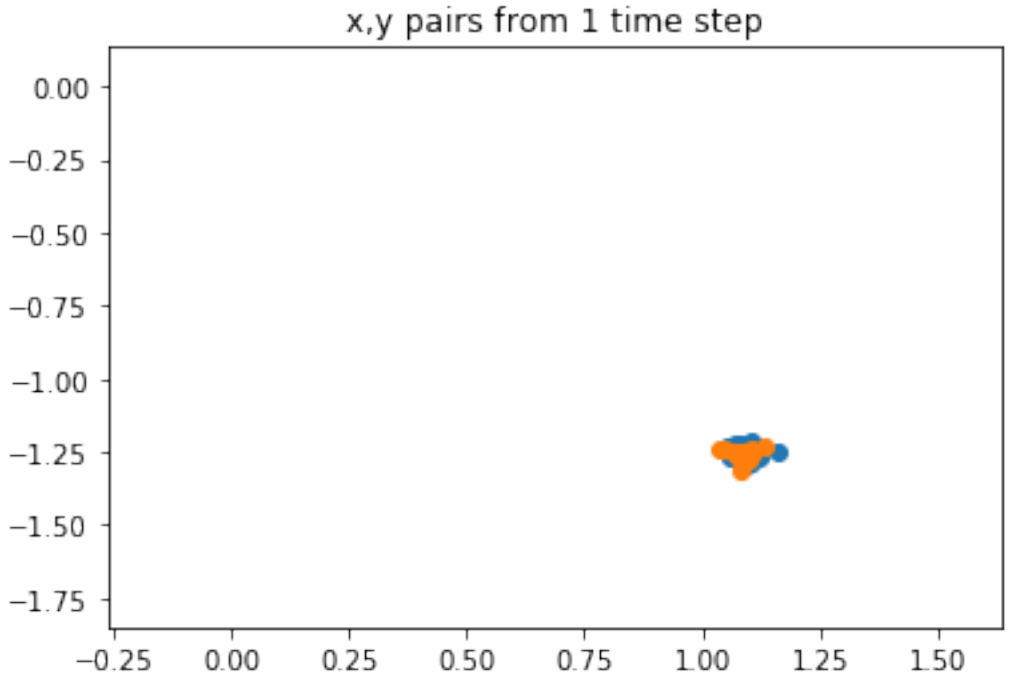
[72]:
```
(3, 399, 40)
```

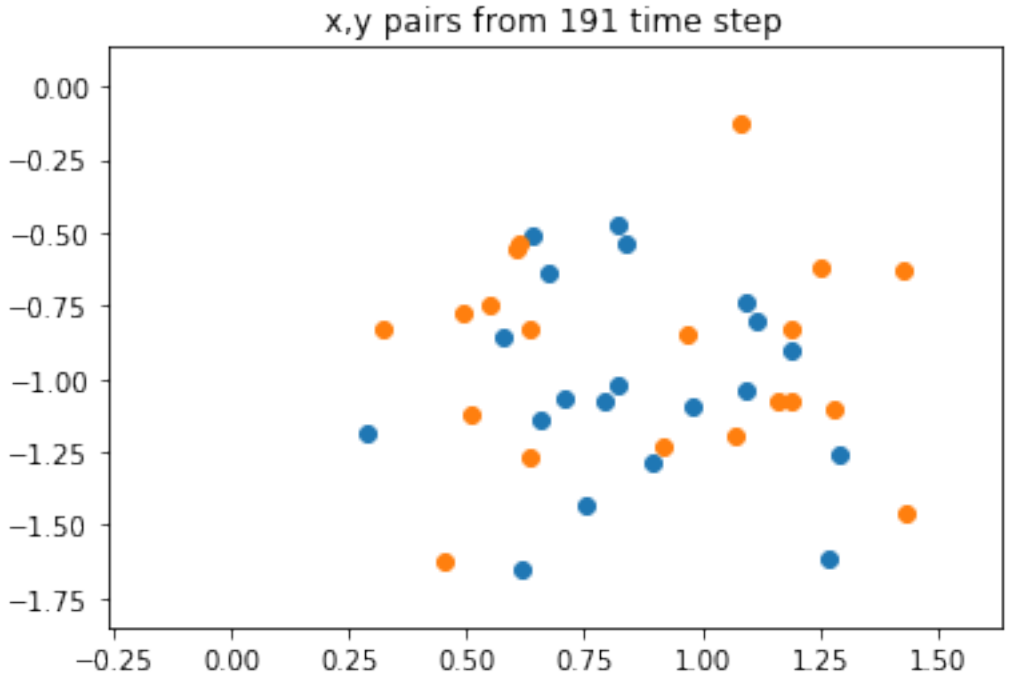
[81]:
```
_, steps, _ = Mt.shape

xMin = None
xMax = None
yMin = None
yMax = None
zMin = None
zMax = None

stepSize = 190

for i in range(0, steps, stepSize):
    G = np.squeeze(Mt[:,i,:])

    if xMin is None or min(G[0,:]) < xMin:
        xMin = min(G[0,:])
    if xMax is None or max(G[0,:]) > xMax:
        xMax = max(G[0,:])

    if yMin is None or min(G[1,:]) < yMin:
        yMin = min(G[1,:])
    if yMax is None or max(G[1,:]) > yMax:
        yMax = max(G[1,:])

    if zMin is None or min(G[2,:]) < zMin:
        zMin = min(G[2,:])
    if zMax is None or max(G[2,:]) > zMax:
        zMax = max(G[2,:])

    paddingX = 0.1 #(abs(min(xMin, xMax))) ** (0.01)
    paddingY = 0.1 #(abs(min(yMin, yMax))) ** (0.01)
    paddingZ = 0.1

    xMin -= paddingX
    xMax += paddingX

    yMin -= paddingY
    yMax += paddingY
```

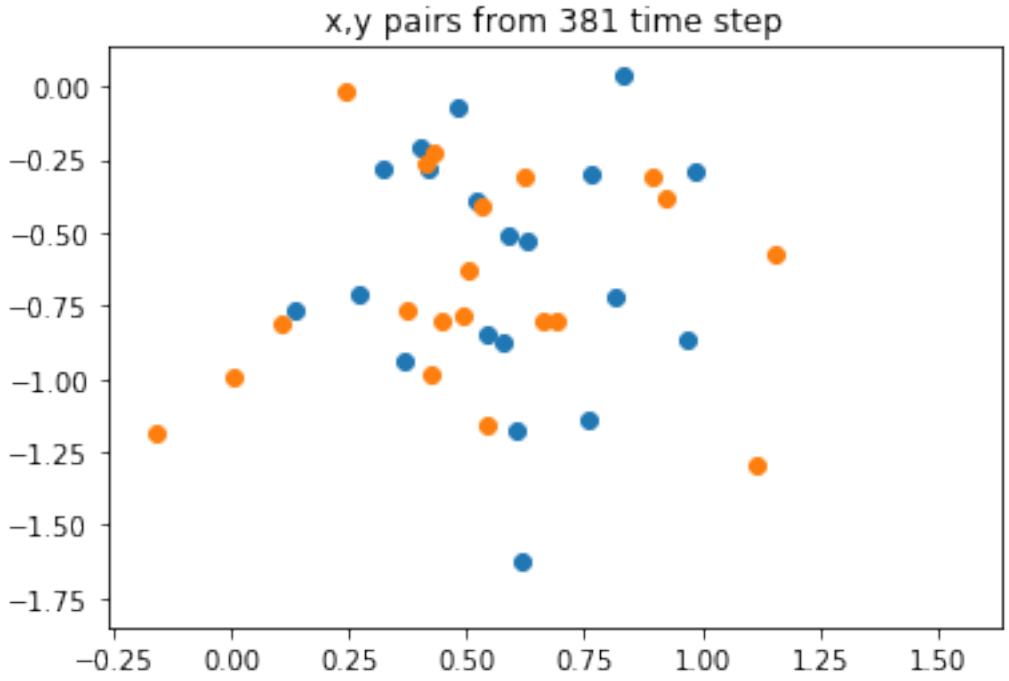
```
    zMin -= paddingZ
    zMax += paddingZ

print("x: {} {} | y: {} {} | z: {} {}".format(xMin, xMax, yMin, yMax, zMin,␣
 ↪zMax))
```

```
x: -0.26189126926585765 1.635521828747712 | y: -1.8503208726016047
0.13633029542345637 | z: -0.16885156143207858 1.825513712429722
```

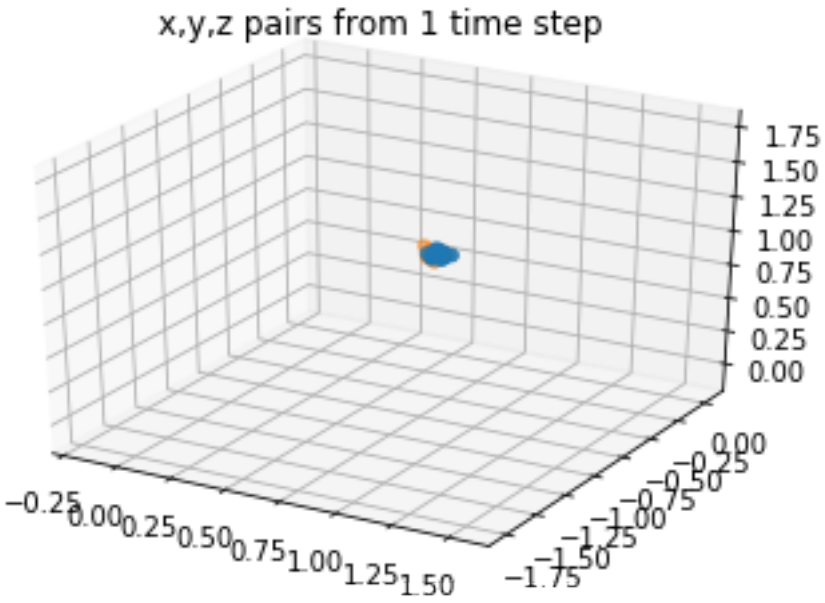
[82]:
```
for i in range(0, steps, stepSize):
    G = np.squeeze(Mt[:,i,:])

    com1pointsX = (G[0,:])[first_community]
    com1pointsY = (G[1,:])[first_community]

    com2pointsX = (G[0,:])[second_community]
    com2pointsY = (G[1,:])[second_community]

    plt.scatter(com1pointsX, com1pointsY)
    plt.scatter(com2pointsX, com2pointsY)

    plt.ylim(yMin, yMax)
    plt.xlim(xMin, xMax)
    plt.title("x,y pairs from {} time step".format(i+1))
    #plt.savefig("{}step.png".format(i))
    plt.show()
```

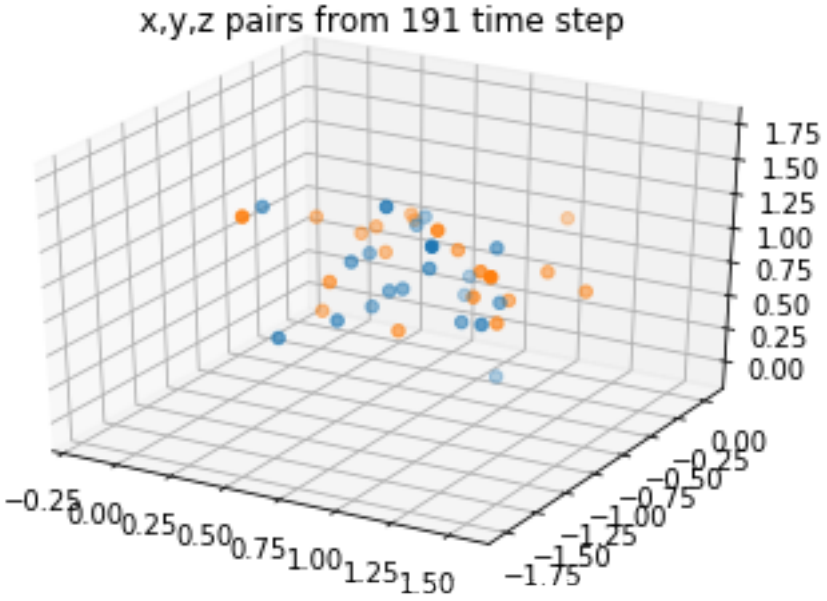


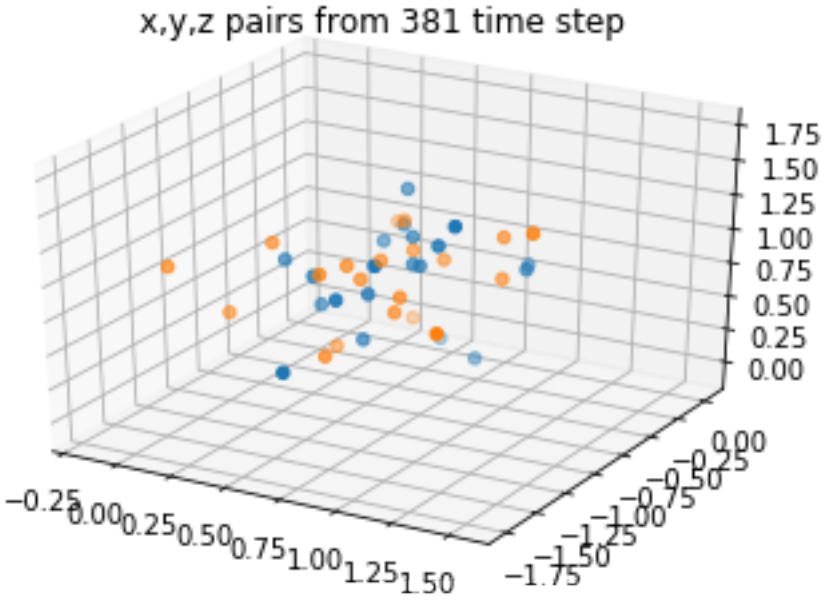

x,y pairs from 191 time step
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
−1.75
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50


x,y pairs from 381 time step
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
−1.75
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50

[83]:
```
_, steps, _ = Mt.shape

for i in range(0, steps, stepSize):
    G = np.squeeze(Mt[:,i,:])

    com1pointsX = (G[0,:])[first_community]
    com1pointsY = (G[1,:])[first_community]
    com1pointsZ = (G[2,:])[first_community]

    com2pointsX = (G[0,:])[second_community]
    com2pointsY = (G[1,:])[second_community]
    com2pointsZ = (G[2,:])[second_community]

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    ax.scatter(com1pointsX, com1pointsY, com1pointsZ)
    ax.scatter(com2pointsX, com2pointsY, com2pointsZ)

    plt.ylim(yMin, yMax)
    plt.xlim(xMin, xMax)
    ax.set_zlim(zMin, zMax)
    plt.title("x,y,z pairs from {} time step".format(i+1))
    #plt.savefig("{}step.png".format(i))
    plt.show()
```




x,y,z pairs from 191 time step
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
−1.75


x,y,z pairs from 381 time step
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
−1.75