ex2

January 3, 2023

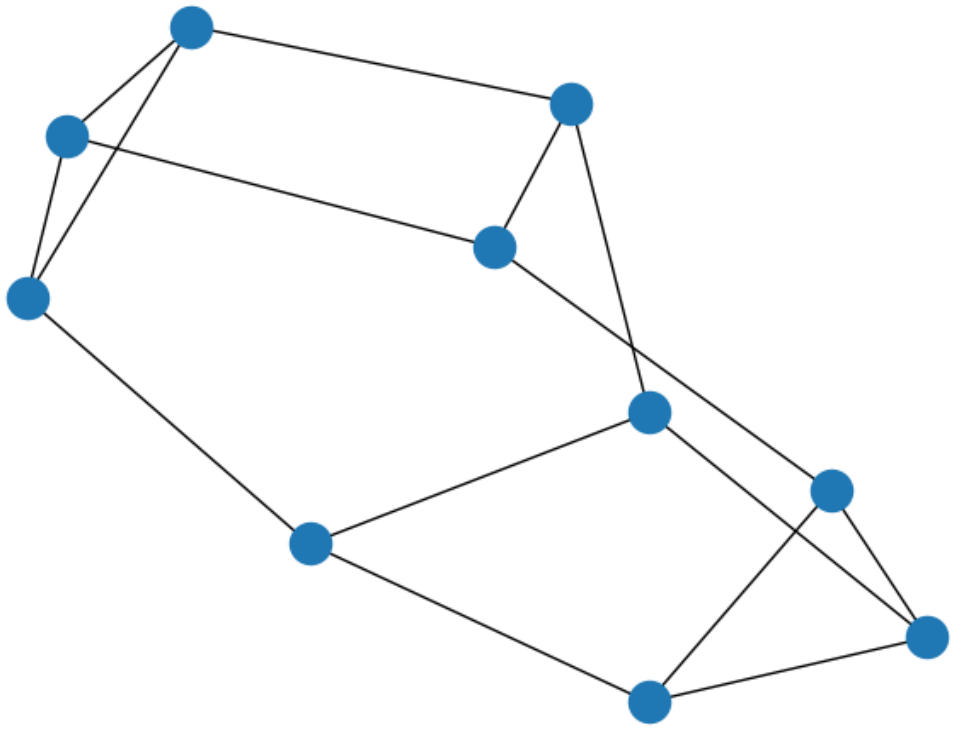
[ ]:
```
import numpy  as np
import networkx as nx
import inspect
```

[ ]:
```
# Generate a regular graph using the network module
G = nx.random_regular_graph(3, 10)
```

[ ]:
```
G.nodes()
```

[ ]:
```
NodeView((4, 9, 3, 7, 6, 1, 0, 8, 2, 5))
```

[ ]:
```
# visualize the graph
nx.draw(G)
```



[ ]:
```
#Generate a laplacian matrix
L = nx.laplacian_matrix(G)
```

[ ]:
```
L.toarray()
```

[ ]:
```
array([[ 3, -1,  0,  0, -1,  0,  0,  0,  0, -1],
       [-1,  3,  0,  0,  0,  0, -1,  0,  0, -1],
       [ 0,  0,  3, -1,  0,  0, -1,  0, -1,  0],
       [ 0,  0, -1,  3,  0, -1,  0, -1,  0,  0],
       [-1,  0,  0,  0,  3, -1,  0,  0, -1,  0],
       [ 0,  0,  0, -1, -1,  3,  0, -1,  0,  0],
       [ 0, -1, -1,  0,  0,  0,  3, -1,  0,  0],
       [ 0,  0,  0, -1,  0, -1, -1,  3,  0,  0],
       [ 0,  0, -1,  0, -1,  0,  0,  0,  3, -1],
       [-1, -1,  0,  0,  0,  0,  0,  0, -1,  3]])
```

[ ]:
```
A = nx.adjacency_matrix(G).toarray()
D = np.diag(np.ones(10)*3)
L1 = D - A
```

```
/tmp/ipykernel_12375/3417495306.py:1: FutureWarning: adjacency_matrix will
return a scipy.sparse array instead of a matrix in Networkx 3.0.
  A = nx.adjacency_matrix(G).toarray()
```

[ ]:
```
L1
```

[ ]:
```
array([[ 3., -1.,  0.,  0., -1.,  0.,  0.,  0.,  0., -1.],
       [-1.,  3.,  0.,  0.,  0.,  0., -1.,  0.,  0., -1.],
       [ 0.,  0.,  3., -1.,  0.,  0., -1.,  0., -1.,  0.],
       [ 0.,  0., -1.,  3.,  0., -1.,  0., -1.,  0.,  0.],
       [-1.,  0.,  0.,  0.,  3., -1.,  0.,  0., -1.,  0.],
       [ 0.,  0.,  0., -1., -1.,  3.,  0., -1.,  0.,  0.],
       [ 0., -1., -1.,  0.,  0.,  0.,  3., -1.,  0.,  0.],
       [ 0.,  0.,  0., -1.,  0., -1., -1.,  3.,  0.,  0.],
       [ 0.,  0., -1.,  0., -1.,  0.,  0.,  0.,  3., -1.],
       [-1., -1.,  0.,  0.,  0.,  0.,  0.,  0., -1.,  3.]])
```

[ ]:
```
np.array_equal(L.toarray(), L1)
```

[ ]:
```
True
```

[ ]:
```
#Ex2.2
# From given Laplace matrix, find the D and A matrix
```

[ ]:
```
L = np.array([[ 2, -1, -1, 0],[-1, 3, -1, -1],[-1, -1, 3, -1],[0, -1, -1, 2]])
L
```

[ ]:
```
array([[ 2, -1, -1,  0],
       [-1,  3, -1, -1],
       [-1, -1,  3, -1],
       [ 0, -1, -1,  2]])
```

[ ]:
```
# find the D matrix
D = np.diag(np.diag(L))
```

[ ]:
```
D
```

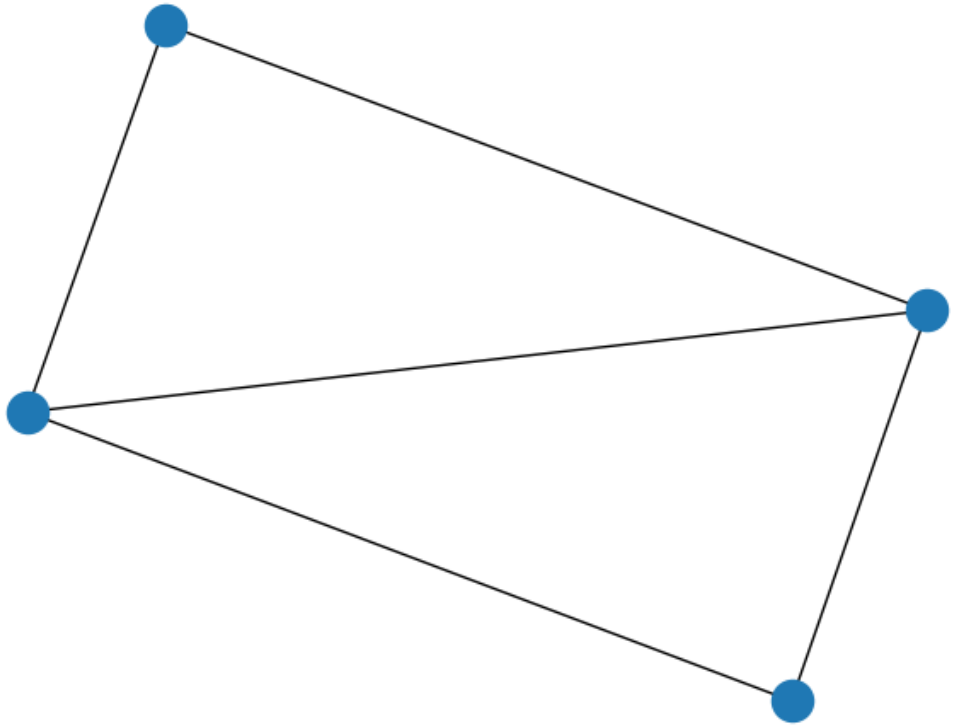
[ ]:
```
array([[2, 0, 0, 0],
       [0, 3, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 2]])
```

[ ]:
```
A = D - L
```

[ ]:
```
A
```

[ ]:
```
array([[0, 1, 1, 0],
       [1, 0, 1, 1],
       [1, 1, 0, 1],
       [0, 1, 1, 0]])
```

[ ]:
```
# plot the graph
G = nx.from_numpy_matrix(A)
nx.draw(G)
```



[ ]: