实验报告

《复杂网络动力学基础》

第二次大作业

姓名：王嘉禾

班级：计算机试验班 001

学号：2193211079

2022 年 10 月 18 日

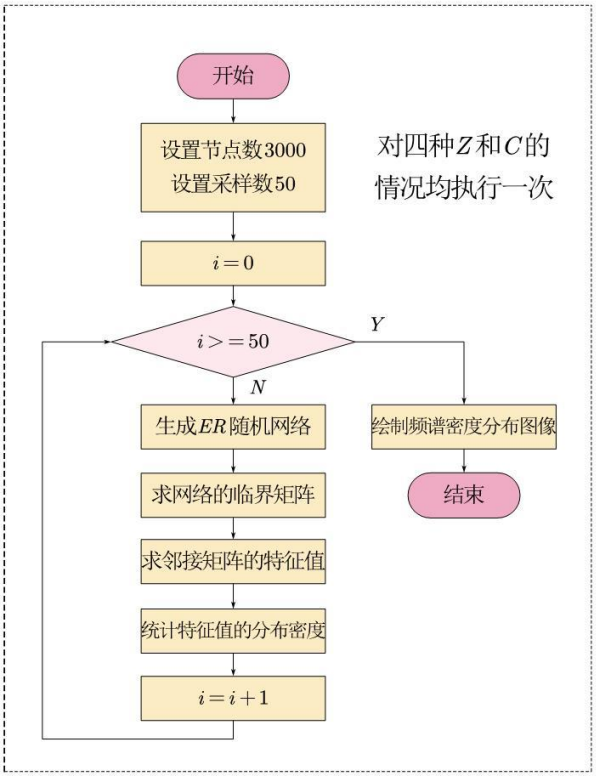
一、题目、问题描述及求解内容(含网络图)

绘制 ER 随机网络 Gn,p 的特征谱，连接概率为 p ，p(N)=cN^(-z)，其中 N 为网络的总节点数，c 和 z 为常数。求出当 N=3000，在以下四种情况 (1)c=0.5，z=1.0 (2)c=1.0，z=1.0 (3)c=1.0，z=1.5 (4)c=10，z=1.0 的谱密度图

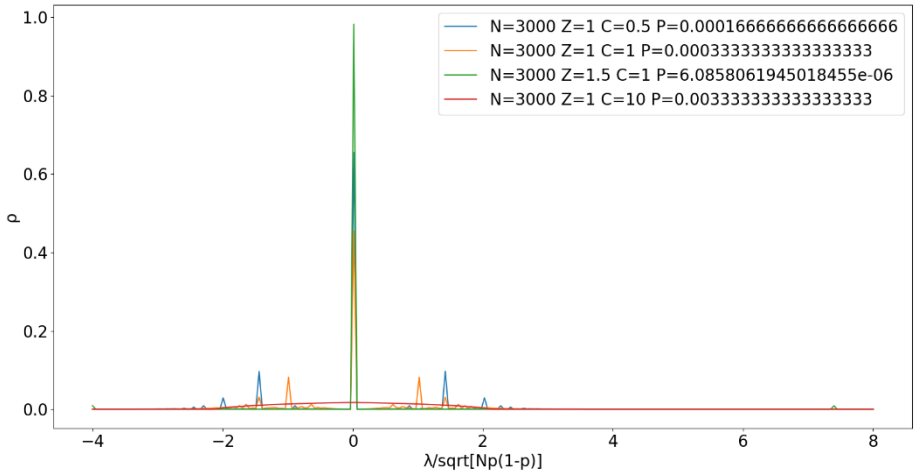
二、实验原理

一个有 N 个节点的网络必有 N 个特征值λ_i，定义其谱密度为$\rho(\lambda) = \frac{1}{N}\sum_{j=1}^{N}\delta(\lambda-\lambda_j)$，其中，分母为网络节点总数，分子为特征值为$\lambda_j$之和，$\rho(\lambda)$表示特征值为$\lambda_j$的概率即谱密度。在作图时，为提高频谱密度的分布状况的可视性，横坐标采用$\lambda / \sqrt{Np(1-p)}$

三、求解过程(含流程图)

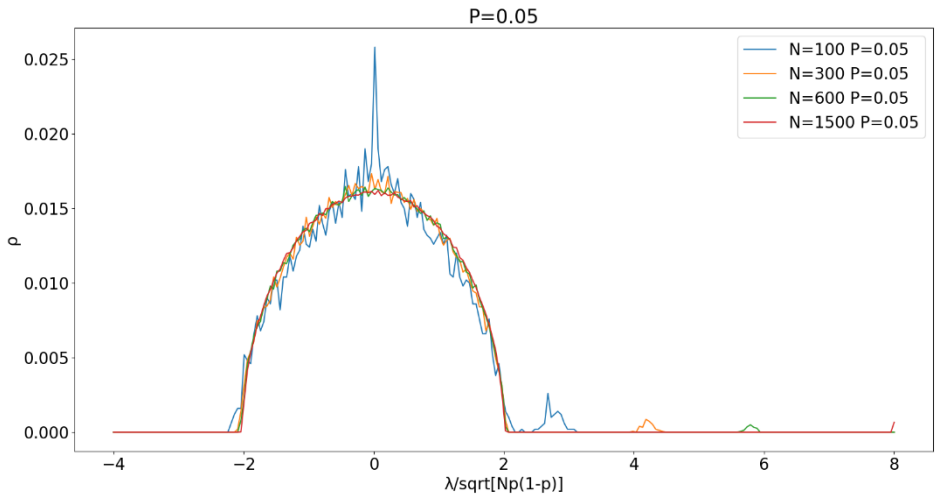


四、实验数据(含表格、曲线、直方图等)



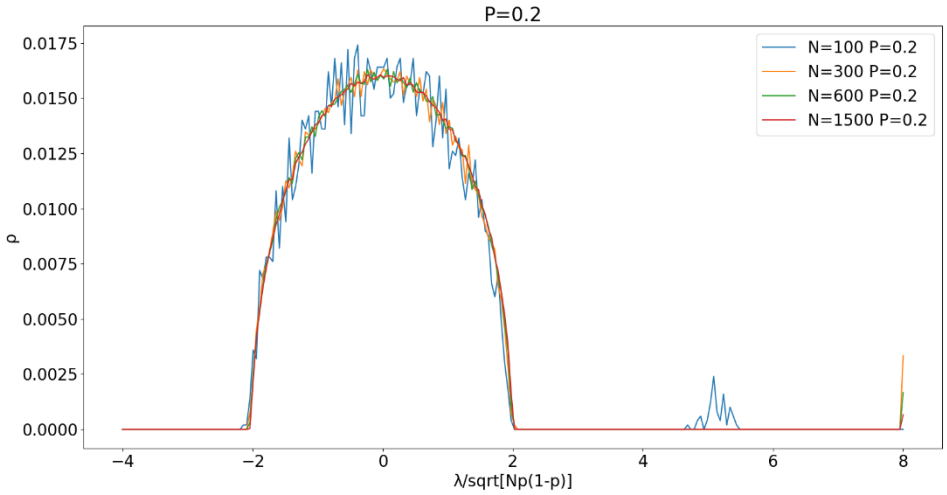
五、实验结果与分析

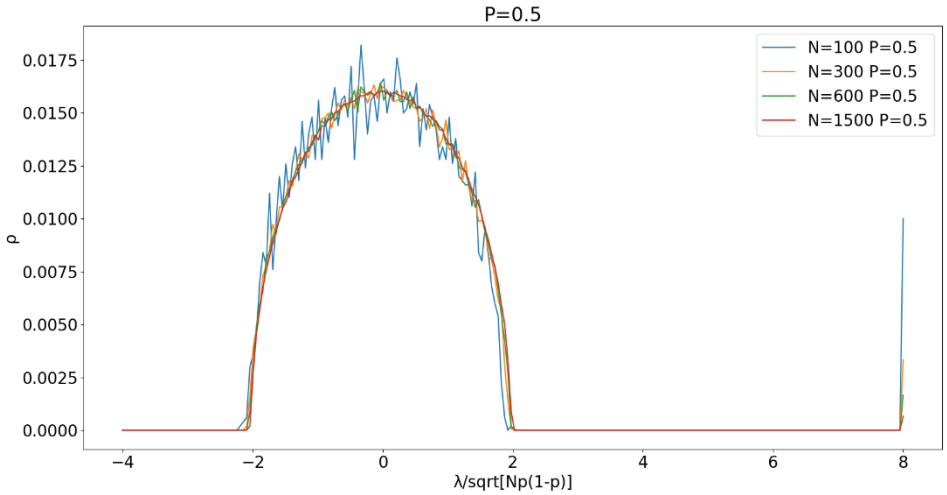
从画出的图像可以看出，频谱密度在前三种情况并不呈现半圆形分布，甚至与半圆形分布相差甚远。但在 Z=10，C=1 时，频谱密度确实呈现半圆形分布。而将链接概率 P 手动设定为 0.05 时，在几种结点个数的情况下，都能明显体现出半圆形分布，并且最大特征值偏离距离中心较远的现象也较明显。



经分析，前三种情况的链接概率都在 1/10000 的数量级，与 0.05 相

差较大，而最后一种情况 Z=10，C=1 时 P=1/300，与 0.05 处在同一数量级。进一步，设置 P 为 0.2 和 0.5，发现半圆形分布的特征越来越明显，波动越来越小，如图所示。





说明链接概率对 ER 随机网络的频谱密度分布存在较大影响，实际上当连接概率很小时（例如本实验中的 1/10000 的数量级），可以看到有大量的 0 特征值，或者说在 0 附近密集分布，这说明网络邻接矩阵的秩很小，这也是符合网络构造的，因为当连接概率很小时，网络的连通性较差，邻接矩阵较稀疏。当链接概率逐渐增大时，频谱密度

分布应趋向于半圆形分布。

六、实验代码、注释及流程图(Matlab、C/C++、Python 等)

```python
import numpy as np
from matplotlib import pyplot
from math import pow
from math import sqrt
from random import random
import networkx as nx
N=3000 #节点数
T=100 #对每一种设定的采样次数

def draw(N,C,Z):
    P=C*(pow(N,-Z))
    Q=sqrt(N*P*(1-P))
    print(P)
    X=np.linspace(-4,8,240) #构造画图的坐标
    Y=np.zeros(len(X))

    for i in range(T):
        G=nx.random_graphs.erdos_renyi_graph(N,P) #生成 ER 随机网络
        A=np.array(nx.adjacency_matrix(G).todense()) #获取邻接矩阵

        eigenvalue,eigenvector=np.linalg.eig(A) #获取频谱
        eigenvalue=eigenvalue/Q #归一化
        tmp=np.zeros(len(X))

        for item in eigenvalue:
            array=np.asarray(X)
            index=(np.abs(array-item)).argmin() #对每个特征值，找到其最接近的横坐标结点
            tmp[index]+=1 #作用相当于计算密度
        tmp=tmp/np.sum(tmp) #求概率，归一化
        Y+=tmp
    Y/=T #多次采样求平均值，归一化
    pyplot.plot(X,Y,label="N="+str(N)+" C="+str(C)+" Z="+str(Z))

draw(N,0.5,1) #四种情况作图
draw(N,1,1)
draw(N,1,1.5)
draw(N,10,1)
pyplot.legend()
pyplot.show()
```

七、实验感想与建议

刚开始试验时，发现绘制的图像不符合半圆形分布，以为是程序出现逻辑错误等，进检查后程序无误，将书中样例带入运行得到了正确结果，后经过思考判断出了出现这种实验结果的原因（在五、实验结果与分析中已经提及）说明做实验时不能硬推结论，应当在对现象的理解基础上再去验证并作出思考，否则可能会出现莫须有的调试困难。

一、 题目、问题描述及求解内容(含网络图)

试用 Python 绘制 WS 小世界网络的度分布 P(k)图。初始网络为规则网络，选取最近邻耦合网络，其中，节点总数 N=1000，耦合数 K=6，n 为未重连的边数，ñ 为重连的边数，随机化重连概率分别为 p=0，0.1，0.2，0.4，0.6，0.9，1.0 。计算公式为：

$$P(k) = \sum_{n=0}^{min\left(k-\frac{K}{2},\frac{K}{2}\right)} C_{\frac{K}{2}}^{n} C_{\frac{K}{2}}^{k-\frac{K}{2}-n} p^{k-2n}(1-p)^{k+2n-k}$$

试分析度分布公式 P(k)，它的未重连边数 n 的求和区间如果选取 max(k-K,K/2)，结果又如何？当 k≥K/2 时，k 最大值可为多少？请给出实验过程、结果及相关分析。要求每个节点的度值 k≥K/2 且保证节点的连通性不被破坏

二、 实验原理(含原理图)

WS 小世界网络的构造是考虑一个耦合度为 K 的最近邻耦合网络，对网络中的边进行“随机化重连”。将图中的每条边以概率 p 随机地重新连接，即将边的一个端点保持不变，而另一个端点以概率 p 与网络的其余 N-K-1 个节点随机连接，且规定：任意两个不同的节点之间至多只能有一条边，即若重连的两个结点之间有边，则该边就不进行链接（边不重复）。

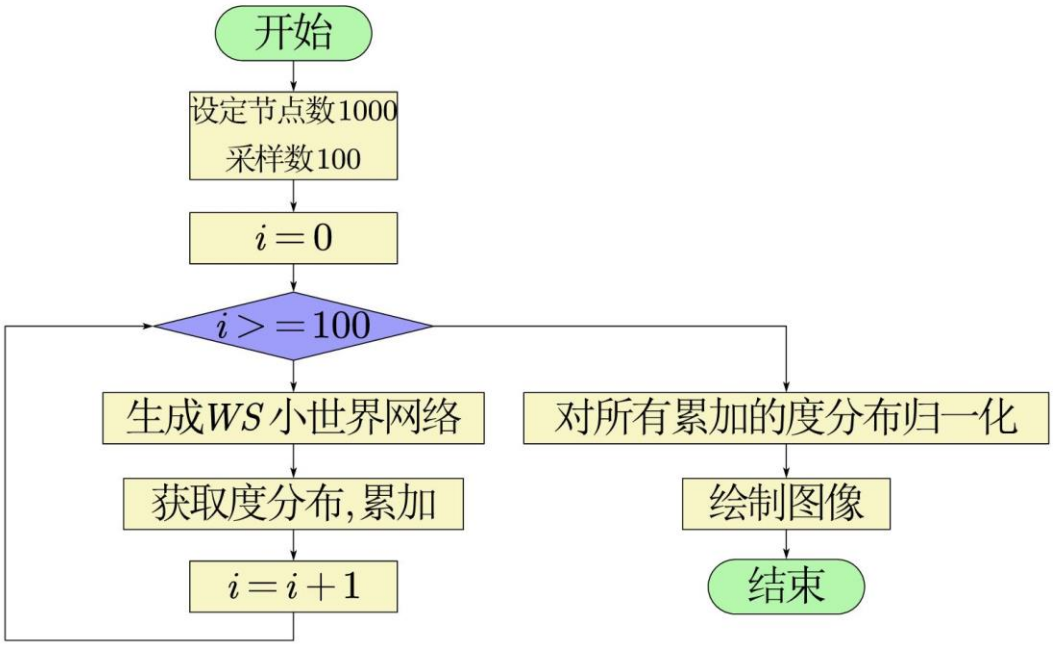
以这种规则构造出的 WS 小世界网络，其节点的度分布可以用一下公式表示：

$$P(k)=\sum_{n=0}^{min\left(k-\frac{K}{2},\frac{K}{2}\right)} C_{\frac{K}{2}}^{n} C_{\frac{K}{2}}^{k-\frac{K}{2}-n} p^{k-2n}(1-p)^{k+2n-k}$$

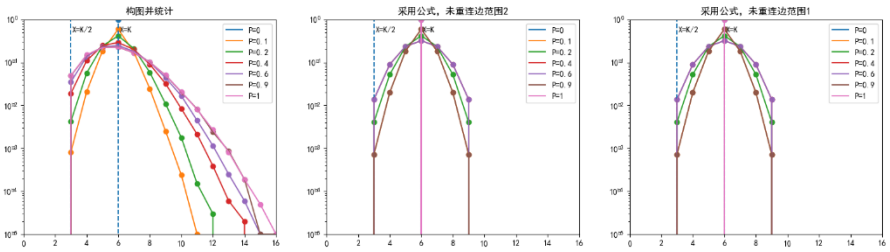
其中 n 表示没有重连的边数，也可以将没有重连的边数范围设置为 $\max\left(k-K,\frac{K}{2}\right)$

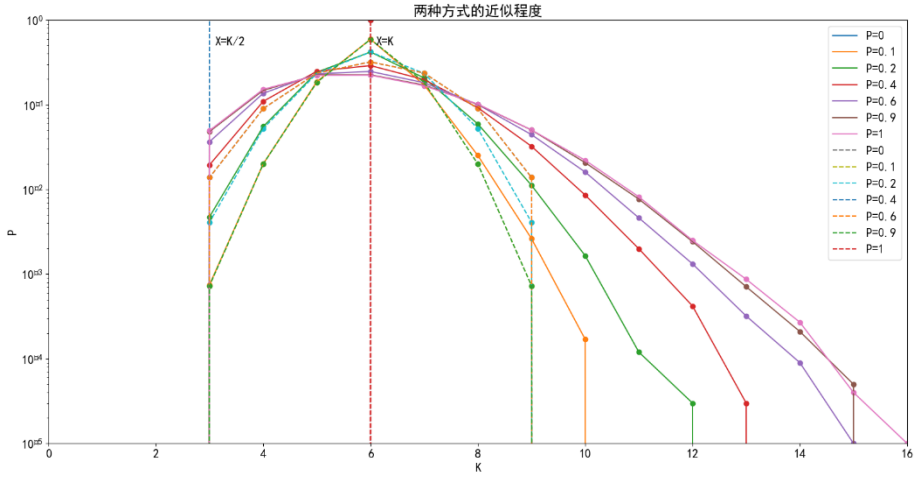
三、 求解过程(含流程图)

使用公式求得度分布的过程过于简单，直接对 k≥K/2 的所有 k 带入公式求得数值并绘图即可，以下给出统计方式求得度分布的流程图



四、 实验数据(含表格、曲线、直方图等)





五、 实验结果与分析

可以看到，以构造网络并统计的方式得到的 WS 小世界网络的度分布与书中给出的结论相符，而以两种公式求出的度分布相同，但体现出局限性：首先是度数被限定在一个范围[K/2,3K/2]中（当 k > K/2 时，k 最大为 3K/2），且度分布关于 K 严格对称，重连概率 P 的情况与 1-P 的情况算出的度分布也完全相同。而在比对两种方法的相似性时，根据实验数据中图二可看出，在该度数范围内，两种方法求出的度分布基本近似，在 K 处几乎完全重合，随着度数偏离 K 的程度误差逐渐增大。

分析：这种现象应是两种方法对“重连”的不同定义造成的，在定义中，重连为完全随机，但在公式中，最近邻耦合网络中每个点的 K 条连边都必须至少保留 K/2 条以保证图的连通性，这会对图最终的度分布产生很大的局限性。其次，这种概率估计是一种近似，对于从自己出发重连还是从其他节点出发重连回自己的两种情况考虑不足，并且也导致了上述两种对称性的出现。

六、实验代码、注释及流程图(Matlab、C/C++、Python 等)

```python
import numpy as np
from matplotlib import pyplot
import networkx as nx
from math import comb

N = 1000
K = 6
KK = 3
pyplot.rcParams.update({'font.size':10})

def draw(N, K, P): #构造网络并统计的方式
    Y = np.zeros((20))
    T = 100 #采样次数
    for i in range(T):
        WSG = nx.random_graphs.watts_strogatz_graph(N, K, P) #生成 WS 小世界网络
        tmp = nx.degree_histogram(WSG) #获取度分布
        tmp /= np.sum(tmp) #归一化
        for j in range(len(tmp)):
            Y[j] += tmp[j]
    Y /= T #多次采样求平均值
    X = np.arange(0, len(Y))
    if (P == 0): #最近邻耦合网络特殊处理
        X = [K]
        Y = [1]
    pyplot.scatter(X, Y)
    pyplot.plot(X, Y, label="P=" + str(P))

def get(N, K, P):
    Y = np.zeros((20))
    X = np.arange(0, len(Y))
    for k in range(KK,20):
        for n in range(max(k-K,KK)+1): #第二种未重连的边的范围情况
            if(k-n-KK<0 or (k-2*n<0 and P==0) or (1-P==0 and K+2*n-k<0)): #删减不符合条件的迭代
                continue
            Y[k]+=comb(KK,n)*comb(KK,k-n-KK)*pow(P,k-2*n)*pow(1-P,K+2*n-k) #调用公式
    if (P == 0):
        X = [K]
        Y = [1]
    pyplot.scatter(X, Y)
```

```
    pyplot.plot(X, Y, label="P=" + str(P))

def get2(N, K, P): #第一种重连的边的范围情况
    Y = np.zeros((20))
    X = np.arange(0, len(Y))
    for k in range(KK,20):
        for n in range(min(k-KK,KK)+1): #第一种未重连的边的范围情况
            if(k-n-KK<0 or (k-2*n<0 and P==0) or (1-P==0 and K+2*n-k<0)):
                continue
            Y[k]+=comb(KK,n)*comb(KK,k-n-KK)*pow(P,k-2*n)*pow(1-P,K+2*n-k)
    if (P == 0):
        X = [K]
        Y = [1]
    pyplot.scatter(X, Y)
    pyplot.plot(X, Y, label="P=" + str(P))
#作图部分
pyplot.subplot(1,3,1)
pyplot.yscale('log')
pyplot.xlim(0, 16)
pyplot.ylim(0.00001, 1)
pyplot.vlines(K, -1, 2, linestyles="dashed")
pyplot.text(K + 0.1, 0.5, "X=K")
pyplot.vlines(K / 2, -1, 2, linestyles="dashed")
pyplot.text(K / 2 + 0.1, 0.5, "X=K/2")
draw(N, K, 0)
draw(N, K, 0.1)
draw(N, K, 0.2)
draw(N, K, 0.4)
draw(N, K, 0.6)
draw(N, K, 0.9)
draw(N, K, 1)
pyplot.legend()

pyplot.subplot(1,3,2)
pyplot.yscale('log')
pyplot.xlim(0, 16)
pyplot.ylim(0.00001, 1)
pyplot.vlines(K, -1, 2, linestyles="dashed")
pyplot.text(K + 0.1, 0.5, "X=K")
pyplot.vlines(K / 2, -1, 2, linestyles="dashed")
pyplot.text(K / 2 + 0.1, 0.5, "X=K/2")
get(N, K, 0)
```

```
get(N, K, 0.1)
get(N, K, 0.2)
get(N, K, 0.4)
get(N, K, 0.6)
get(N, K, 0.9)
get(N, K, 1)
pyplot.legend()

pyplot.subplot(1,3,3)
pyplot.yscale('log')
pyplot.xlim(0, 16)
pyplot.ylim(0.00001, 1)
pyplot.vlines(K, -1, 2, linestyles="dashed")
pyplot.text(K + 0.1, 0.5, "X=K")
pyplot.vlines(K / 2, -1, 2, linestyles="dashed")
pyplot.text(K / 2 + 0.1, 0.5, "X=K/2")
get2(N, K, 0)
get2(N, K, 0.1)
  get2(N, K, 0.2)
  get2(N, K, 0.4)
  get2(N, K, 0.6)
  get2(N, K, 0.9)
  get2(N, K, 1)
  pyplot.legend()
  pyplot.show()
```

七、实验感想与建议

这种近似的概率估计有很大的局限性，因为从未重连边数和重连边数的概率公式来看，重连边并未明确考虑在其他节点重连并连回该结点的情况，或者只作出的估计。并且，保留 K/2 条边的操作是否多余也有待商榷，应当深入研究并给出更符合直觉的概率估计公式。