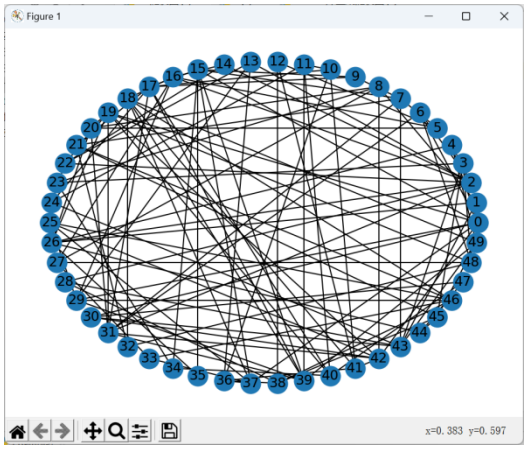
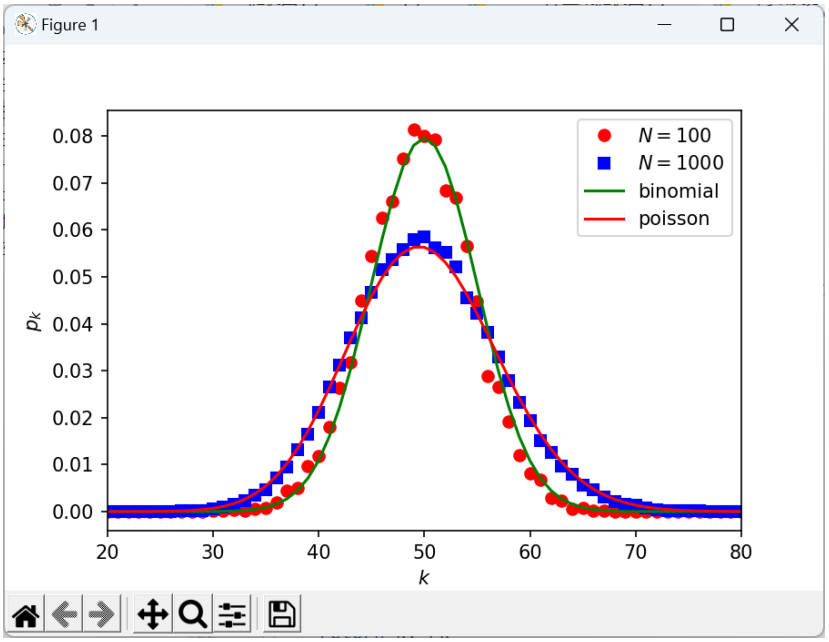
```python
import random
import networkx as nx
import matplotlib.pyplot as plt

def GNM(N,M):
    G = nx.Graph()
    G.add_nodes_from(range(N))
    nlist = list(G)
    edge_count = 0
    while edge_count < M:
        u = random.choice(nlist)
        v = random.choice(nlist)
        if u == v or G.has_edge(u,v):
            continue
        else:
            G.add_edge(u,v)
            edge_count += 1
    return G

G = GNM(50,125)

nx.draw(G, with_labels=True, pos=nx.circular_layout(G))
plt.show()
```

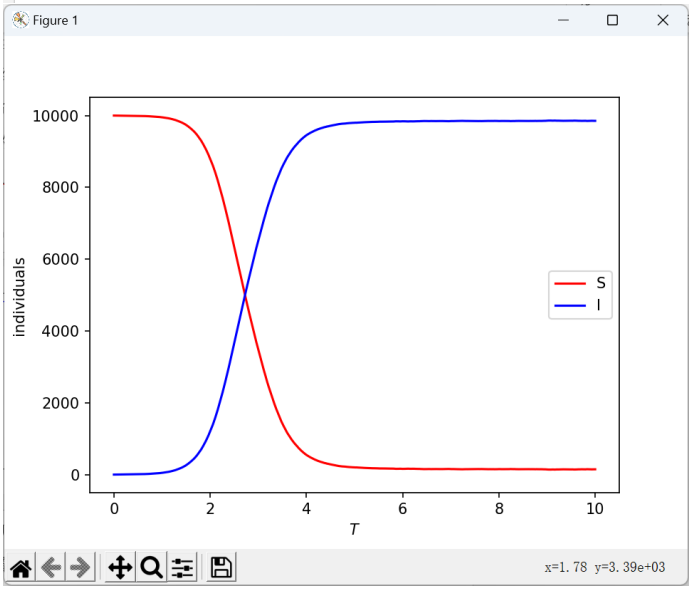
```python
import random
import itertools
import networkx as nx
import matplotlib.pyplot as plt

def GNP(N,p):
    edges = itertools.combinations(range(N),2)
    G = nx.Graph()
    G.add_nodes_from(range(N))
    for e in edges:
        if random.random() < p:
            G.add_edge(*e)
    return G

G = GNP(50,0.1)

nx.draw(G, with_labels=True, pos=nx.circular_layout(G))
plt.show()
```




```python
samples = 100  # 统计平均
N = [100, 1000]
# 为了便于统计平均，指定区间[20,80]
kmin, kmax, avk = 20, 80, 50
s1 = np.zeros(kmax - kmin + 1)
s2 = np.zeros(kmax - kmin + 1)
for i in range(samples):
    ER1 = nx.gnp_random_graph(N[0], avk / N[0])
    x1, y1 = get_pdf(ER1, kmin, kmax)
    ER2 = nx.gnp_random_graph(N[1], avk / N[1])
    x2, y2 = get_pdf(ER2, kmin, kmax)

    s1 += np.array(y1)
    s2 += np.array(y2)
```




```python
def monte_carlo_simulation(ER, tau, gamma, num_simulations, tmax):
    results = []
    for _ in range(num_simulations):
        t, S, I = EoN.fast_SIS(ER, tau=tau, gamma=gamma, tmax=tmax)
        results.append((t, S, I))
    return results

def average_results(results):
    max_t = max(result[0][-1] for result in results)
    avg_t = np.linspace(0, max_t, len(results[0][0]))  # 使用线性插值得到相同长度的时间数组
    avg_S = np.mean([np.interp(avg_t, result[0], result[1]) for result in results], axis=0)
    avg_I = np.mean([np.interp(avg_t, result[0], result[2]) for result in results], axis=0)
    return avg_t, avg_S, avg_I

N = 10**4
M = 4 * N
ER = nx.gnm_random_graph(N, M)

tau = 0.5          # transmission rate
gamma = 0.05        # recovery rate per node
num_simulations = 20
tmax = 10

results = monte_carlo_simulation(ER, tau, gamma, num_simulations, tmax)
avg_t, avg_S, avg_I = average_results(results)
```



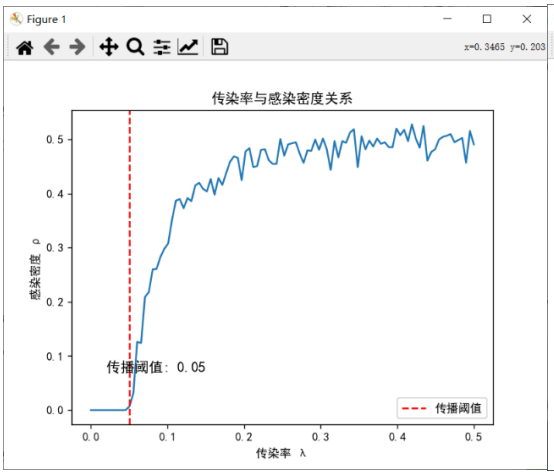


Figure 1
x=0.3465 y=0.203
传染率与感染密度关系
感染密度 ρ
传播阈值：0.05
传播阈值
传染率 λ

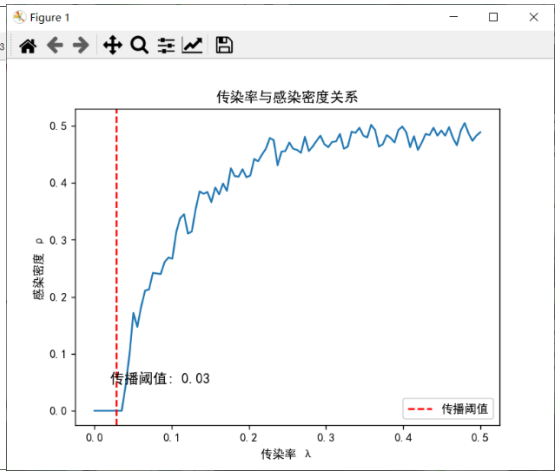


Figure 1
传染率与感染密度关系
感染密度 ρ
传播阈值：0.03
传播阈值
传染率 λ