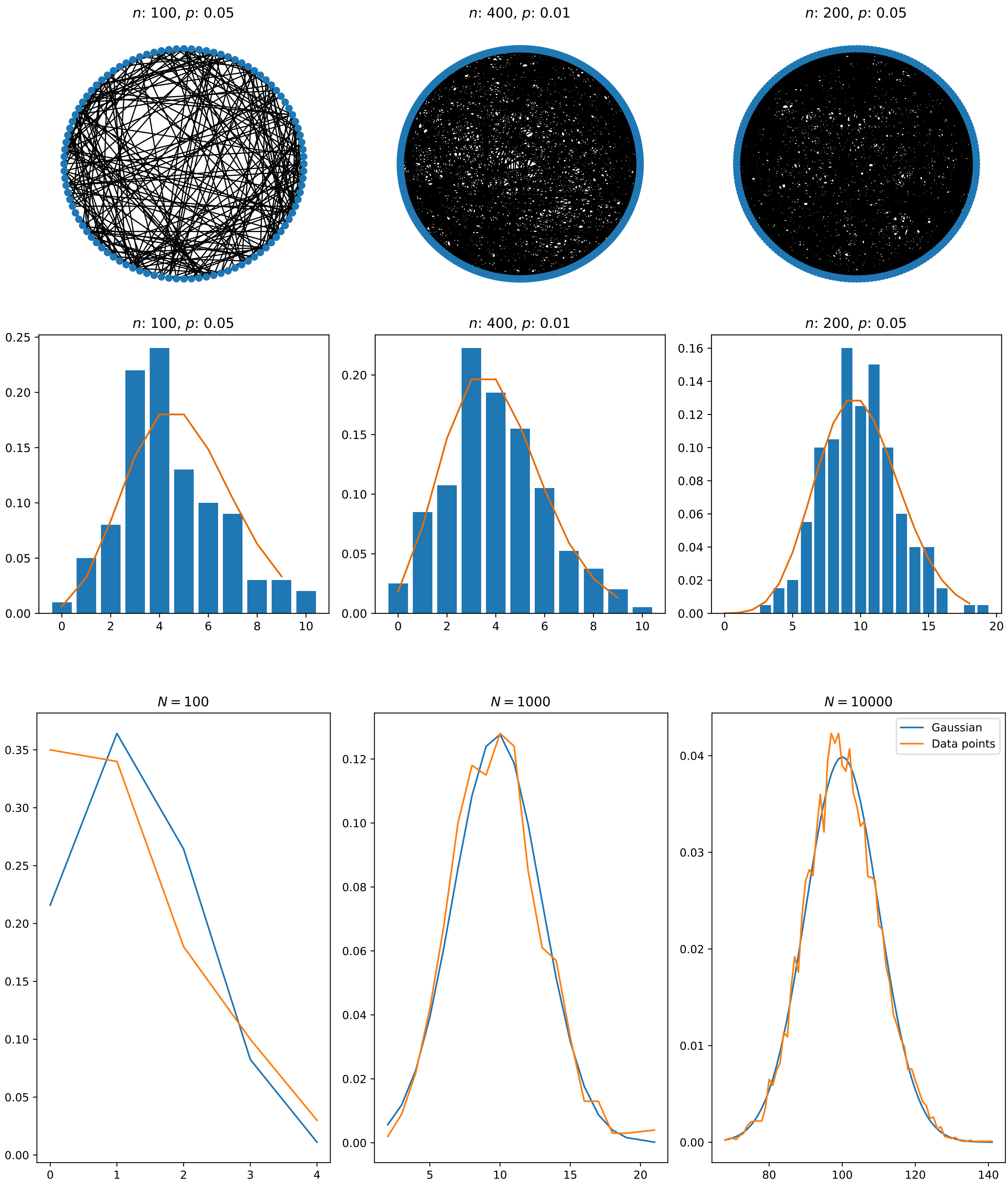
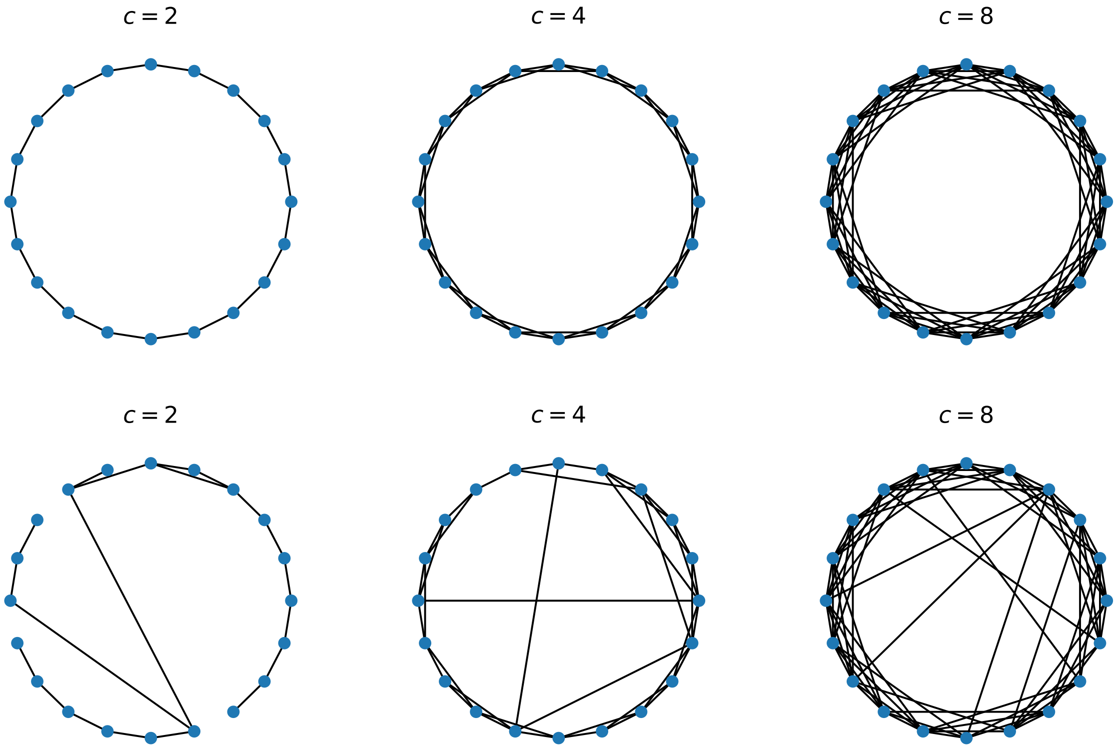
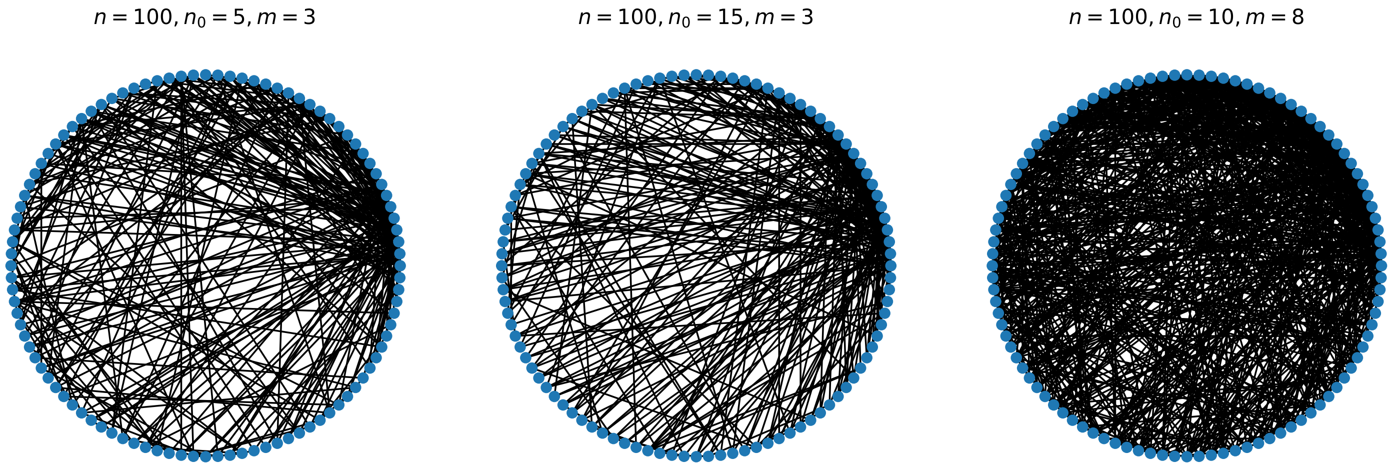
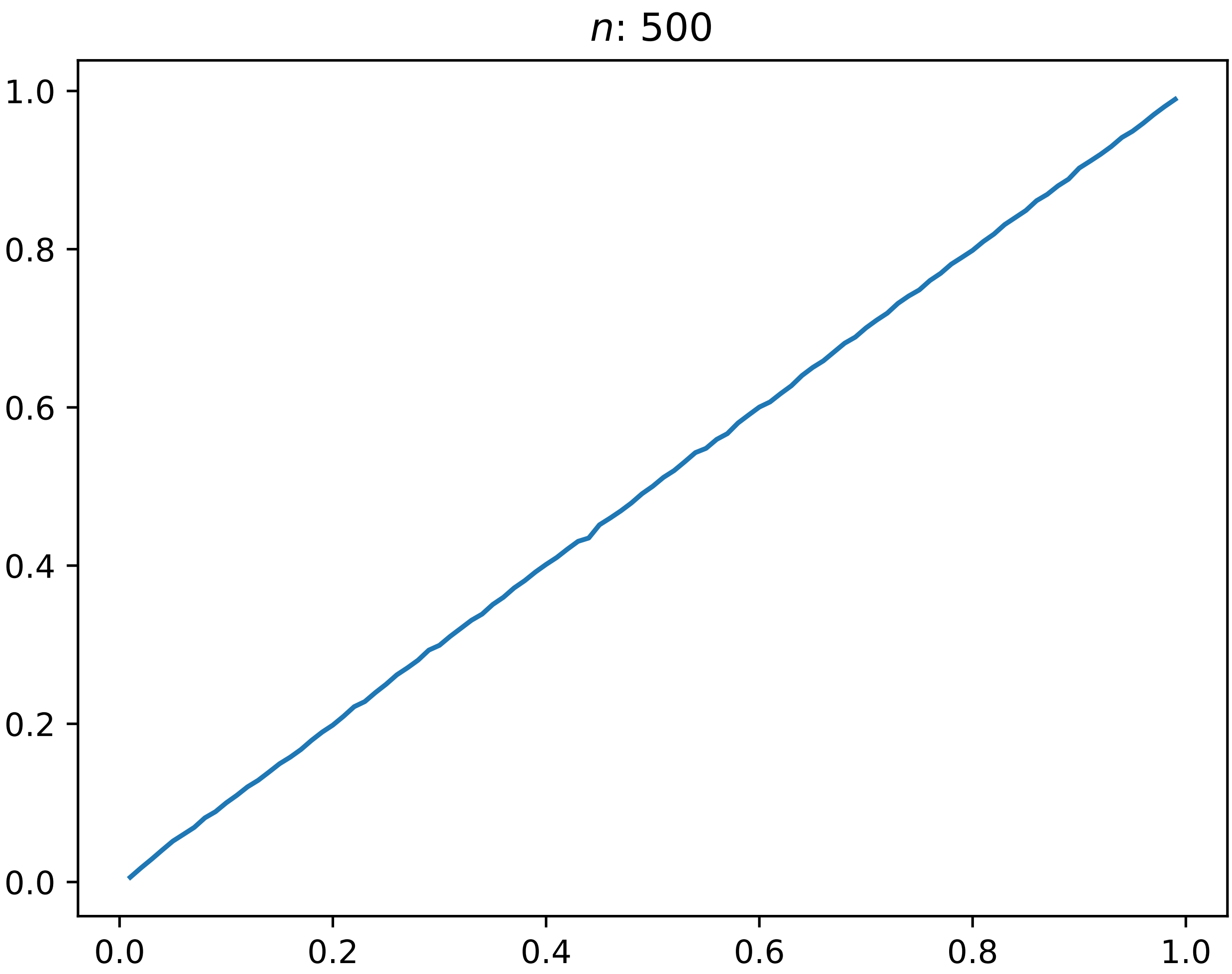

n: 100, p: 0.05
n: 400, p: 0.01
n: 200, p: 0.05
n: 100, p: 0.05
n: 400, p: 0.01
n: 200, p: 0.05
N = 100
N = 1000
N = 10000
Gaussian
Data points


$c = 2$
$c = 4$
$c = 8$
$c = 2$
$c = 4$
$c = 8$


$n = 100, n_0 = 5, m = 3$
$n = 100, n_0 = 15, m = 3$
$n = 100, n_0 = 10, m = 8$



/Erdos-Renyi/main.py

```
#!/usr/bin/python

import numpy as np
import networkx
import scipy

from matplotlib import pyplot as plt

def gen_graph(n, p):
    A = np.zeros((n,n))
    for i in range(n):
        for j in range(i+1,n):
            A[i,j] = np.random.rand()<p
            A[j,i] = A[i,j]

    return A

def degree_prob(n, k, p):
    return scipy.special.comb(n-1, k) * np.power(p, k) * np.power((1 - p),n-1-k)

def get_degrees(A, return_connetions=False):
    n = A.shape[0]
    connections = np.zeros(n)
    for j in range(n):
        connections[j] = np.count_nonzero(A[j,:])
    if return_connetions:
        (i,x) = np.unique(connections, return_counts=True)
        return (i, x, connections)
    else:
```

```
        return np.unique(connections, return_counts=True)

if __name__ == "__main__":
    for (i, n, p) in [(1, 100, 0.05), (2, 400, 0.01), (3, 200, 0.05)]:
        A = gen_graph(n, p)

        G = networkx.from_numpy_array(A)

        plt.subplot(2, 3, i)
        plt.title(f"$n$: {n}, $p$: {p}")
        pos = networkx.circular_layout(G)
        networkx.draw(G, pos=pos, node_size=30)

        plt.subplot(2, 3, 3+i)
        plt.title(f"$n$: {n}, $p$: {p}")

        k, k_count = get_degrees(A)
        plt.bar(k, k_count / n)
        plt.plot(np.arange(np.max(k)), degree_prob(n, np.arange(np.max(k)), p), color="#
            E66C00")

    plt.show()
```

/Erdos-Renyi/gaussian.py

```
import numpy as np

from main import *
from matplotlib import pyplot as plt

def normal_dist(x, mu=0.0, sigma=1.0):
    return 1/(sigma*np.sqrt(2*np.pi)) * np.exp(-0.5*((x-mu)/sigma)**2)

if __name__ == "__main__":
    for (i, N) in enumerate([100, 1000, 10000]):
        A = gen_graph(N, 0.01)

        x = np.linspace(-10, 10, num=N)

        (k, k_count, connections) = get_degrees(A, return_connetions=True)
        plt.subplot(1,3,i+1)
        plt.plot(k, normal_dist(k, mu=np.mean(connections), sigma=np.std(connections)), label
            ="Gaussian")
        plt.plot(k, k_count/N, label="Data points")
        plt.title(f"$N = {N}$")

    plt.legend()
    plt.show()
```

/Watts-Strogatz/main.py

```
#!/usr/bin/python

import numpy as np

import networkx as nx
import random


from matplotlib import pyplot as plt

def gen_matrix(n, c, p):
    A = np.zeros((n,n))
    for i in range(n):
        for j in range(int(c/2)):
            A[i,(i+j+1)%n] = 1
            if np.random.rand()<p:
                rewire_index = random.sample(list(np.where(np.logical_not(A[i,:])&np.
                    logical_not(range(n)==i))[0]),1)
                A[i,(i+j+1)%n] = 0
                A[i,rewire_index] = 1
    A = A + np.transpose(A)
    return(A)


if __name__ == "__main__":

    for i, (c, p) in enumerate([(2, 0.0), (4, 0.0), (8, 0.0), (2, 0.2), (4, 0.2), (8, 0.2)]):
        A = gen_matrix(20, c, p)

        plt.subplot(2,3,i+1)
        plt.title(f"$c = {c}$")
        G = nx.from_numpy_array(A)
        pos = nx.circular_layout(G)
        nx.draw(G, pos, node_size=30)

    plt.show()
```

/Albert-Barabasi/main.py

```
import networkx as nx
import numpy as np

from matplotlib import pyplot as plt

from random import sample

def preferentialgrowth_graph(n,n0,m):
    A = np.zeros((n,n))
    for i in range(n0):
        for j in range(i+1,n0):
            A[i,j] = 1
    A = A + np.transpose(A)
    for t in range(n-n0):
        D = np.sum(A, axis=0)/np.sum(A)
        edges = np.random.choice(np.arange(n), m ,replace=False, p=D)
        for i in range(m):
            A[t+n0,edges[i]] = 1
            A[edges[i],t+n0] = 1

    return(A)

if __name__ == "__main__":
    for i, n, n0, m in [(1, 100, 5, 3), (2, 100, 15, 3), (3, 100, 10, 8)]:
        A = preferentialgrowth_graph(n, n0, m)

        G = nx.from_numpy_array(A)

        plt.subplot(1, 3, i)
        plt.title(f"$n = {n}, n_0 = {n0}, m = {m}$")
        pos = nx.circular_layout(G)
        nx.draw(G, pos=pos, node_size=30)

    plt.show()
```

/Albert-Barabasi/dist.py

```
import networkx as nx
import numpy as np

from matplotlib import pyplot as plt

from main import preferentialgrowth_graph

if __name__ == "__main__":
    n  = 1000
    m  = 3
    n0 = 5

    A = preferentialgrowth_graph(n, n0, m)
    degrees = np.sum(A,1)

    plt.loglog(np.sort(degrees)[::-1],np.arange(n)/n,'.')
    plt.loglog(np.arange(m,np.max(degrees)), m**2 * np.arange(m,np.max(degrees))**(-2),'--')

    plt.title("Degree distribution")
    plt.ylabel('$C(k)$')
    plt.xlabel('$k$')

    plt.show()
```

/Average-Erdos-Renyi/gaussian.py

```
#!/usr/bin/python

import numpy as np
import networkx as nx
import scipy

from matplotlib import pyplot as plt

def gen_graph(n, p):
    A = np.zeros((n,n))
    for i in range(n):
        for j in range(i+1,n):
            A[i,j] = np.random.rand()<p
            A[j,i] = A[i,j]

    return A

def degree_prob(n, k, p):
    return scipy.special.comb(n-1, k) * np.power(p, k) * np.power((1 - p),n-1-k)

def get_degrees(A, return_connetions=False):
    n = A.shape[0]
    connections = np.zeros(n)
    for j in range(n):
        connections[j] = np.count_nonzero(A[j,:])
    if return_connetions:
        (i,x) = np.unique(connections, return_counts=True)
        return (i, x, connections)
    else:
        return np.unique(connections, return_counts=True)

if __name__ == "__main__":
    n = 500

    avg_length = np.zeros(100)
    cluster = np.zeros(100)
    for (i, p) in enumerate(np.arange(0,100) / 100.0):
        print(i)
        A = gen_graph(n, p)
        A_triple = (A @ A) @ A

        degrees = np.sum(A, axis=0)

        cluster[i] = np.sum(np.trace(A_triple)) / np.sum(degrees * (degrees - 1))

    print(cluster)
    plt.title(f"$n$: {n}")
    plt.plot(np.arange(100) / 100.0, cluster)
    plt.show()
```