**Code for small-scale data generation**

import networkx as nx  
import numpy as np  
import matplotlib.pyplot as plt  
from scipy.interpolate import interp1d  
  
plt.rcParams['font.family'] = 'Arial'  
  
# Generate a random graph  
num\_nodes = 200  
num\_edges = 1200  
G = nx.gnm\_random\_graph(num\_nodes, num\_edges)  
  
# Calculate the initial Clustering Coefficient  
initial\_cc = nx.average\_clustering(G)  
  
# Removal ratio values  
removal\_ratios = np.linspace(0.00, 1.0, num=100)  
  
# Calculate Clustering Coefficient for different removal ratios  
cc\_values = []  
for ratio in removal\_ratios:  
 num\_removal = int(num\_edges \* ratio)  
 edges = list(G.edges())  
 np.random.shuffle(edges)  
 removed\_edges = edges[:num\_removal]  
 G\_removed = G.copy()  
 G\_removed.remove\_edges\_from(removed\_edges)  
 cc = nx.average\_clustering(G\_removed)  
 cc\_values.append(cc)  
  
# Perform piecewise linear interpolation  
interp\_func = interp1d(removal\_ratios, cc\_values, kind='linear')  
  
# Plot Clustering Coefficient vs Removal Ratio with piecewise linear fit  
plt.plot(removal\_ratios, cc\_values, color='steelblue', marker='o', markersize=4, label='Data Points')  
plt.plot(removal\_ratios, interp\_func(removal\_ratios), color='orange', linestyle='-', linewidth=2, label='Piecewise Linear Fit')  
plt.xlabel('Removal Ratio')  
plt.ylabel('Clustering Coefficient')  
plt.title('Change of CC after removing edges according to a certain percentage')  
plt.xticks(np.arange(0, 1.1, 0.1))  
plt.grid(True)  
plt.legend()  
  
  
plt.show()