This implementation analyzes the degree correlation k_nn(k) of a real-world email network and compares it with an ensemble of randomized networks generated using the configuration model. It first computes the average neighbor degree for each degree k in the real graph. Then, it constructs 100 configuration model graphs by randomly rewiring edges while preserving the degree sequence. The k_nn(k) values are averaged across all random graphs and plotted alongside the real graph to study degree correlations.

In [2]:
```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
from collections import defaultdict
from tqdm import tqdm  # Optional: for progress tracking

# Load the graph
file_path = 'email-Eu-core.txt'
G = nx.read_edgelist(file_path, nodetype=int)
print(f"Loaded graph with {G.number_of_nodes()} nodes and {G.number_of_ed

def knn_curve(G):
    """Calculate average neighbor degree for each degree k"""
    knn = nx.average_neighbor_degree(G)
    by_k = defaultdict(list)
    for n, d in G.degree():
        by_k[d].append(knn[n])
    ks = sorted(by_k)
    return ks, [np.mean(by_k[k]) for k in ks]

def configuration_model(G):
    """Generate a random graph with the same degree sequence"""
    stubs = []
    for n, d in G.degree():
        stubs += [n]*d
    np.random.shuffle(stubs)
    H = nx.Graph()
    H.add_nodes_from(G.nodes())
    while stubs:
        # If only one stub remains, break to avoid index error
        if len(stubs) < 2:
            break
        u = stubs.pop(); v = stubs.pop()
        if u != v:  # Avoid self-loops
            H.add_edge(u, v)
    return H

# Compute KNN vs k for the real-world network
print("Computing KNN vs k for the real-world network...")
ks_real, knn_real = knn_curve(G)

# Print the real-world network KNN values
print("\nReal-world network KNN vs k:")
print(f"{'Degree k':<10} {'Average Neighbor Degree knn(k)':<25}")
print("-" * 35)
for k, knn in zip(ks_real, knn_real):
    print(f"{k:<10} {knn:<25.4f}")

# Random config model average
```

```python
print("\nGenerating random graphs and computing average KNN...")
instances = 100
sum_knn = defaultdict(float)
count_knn = defaultdict(int)

for i in tqdm(range(instances), desc="Random Graphs"):
    H = configuration_model(G)
    ks_h, knn_h = knn_curve(H)
    for k, val in zip(ks_h, knn_h):
        sum_knn[k] += val
        count_knn[k] += 1

ks_conf = sorted(sum_knn)
knn_conf = [sum_knn[k]/count_knn[k] for k in ks_conf]

# Print the configuration model average KNN values
print("\nConfiguration model average KNN vs k (100 instances):")
print(f"{'Degree k':<10} {'Average Neighbor Degree knn(k)':<25}")
print("-" * 35)
for k, knn in zip(ks_conf, knn_conf):
    print(f"{k:<10} {knn:<25.4f}")

# Plot the results
plt.figure(figsize=(10, 7))
plt.loglog(ks_real, knn_real, 'o-', label='Real graph', markersize=8)
plt.loglog(ks_conf, knn_conf, 's--', label='Config-model avg', markersize
plt.xlabel('Degree $k$', fontsize=14)
plt.ylabel(r'Average neighbor degree $k_{nn}(k)$', fontsize=14)
plt.legend(fontsize=12)
plt.title('Degree Correlations', fontsize=16)
plt.grid(True, alpha=0.3)
plt.tight_layout()

# Save the figure (optional)
plt.savefig('knn_correlations.png', dpi=300)
plt.show()

# Calculate correlation coefficient (assortativity coefficient)
r = nx.degree_assortativity_coefficient(G)
print(f"\nAssortativity coefficient for real network: {r:.4f}")

# Calculate average assortativity for random networks
random_assort = []
for i in tqdm(range(instances), desc="Random Assortativity"):
    H = configuration_model(G)
    r_h = nx.degree_assortativity_coefficient(H)
    random_assort.append(r_h)

print(f"Average assortativity coefficient for random networks: {np.mean(r
```

```
Loaded graph with 1005 nodes and 16706 edges
Computing KNN vs k for the real-world network...

Real-world network KNN vs k:
Degree k   Average Neighbor Degree knn(k)
----------------------------------
1          71.8846
2          55.5000
3          61.2778
4          65.7121
5          67.9643
6          69.1508
7          69.8114
8          75.0563
9          83.3185
10         76.1214
11         69.2364
12         54.9611
13         63.6442
14         70.4592
15         64.6810
16         59.9010
17         59.5948
18         80.1852
19         63.8702
20         74.7694
21         62.1079
22         70.4403
23         71.6187
24         70.9271
25         64.9422
26         65.3764
27         62.8519
28         64.4375
29         78.7638
30         72.9246
31         75.1056
32         74.7031
33         75.9053
34         80.7157
35         59.5964
36         69.0707
37         72.3661
38         78.3596
39         77.3718
40         64.3250
41         70.3740
42         78.5744
43         82.9103
44         67.4188
45         71.8519
46         74.0242
47         70.0559
48         70.6071
49         80.1714
50         85.0267
51         75.0980
52         86.0048
53         76.2830
54         88.1958
```

55	67.8864
56	78.6679
57	84.8202
58	78.4713
59	73.3277
60	72.5125
61	78.1574
62	74.3806
63	84.2328
64	69.0469
65	89.1923
66	89.4545
67	73.8209
68	78.5980
69	85.3841
70	74.4000
71	76.1662
72	74.0833
73	83.7534
74	81.2050
75	90.7600
76	68.2807
77	78.7922
78	72.8718
79	49.4430
80	80.8542
81	69.9259
82	92.3415
83	80.9253
84	84.7937
85	76.0588
86	73.5378
87	76.7644
88	94.0341
89	82.1798
91	69.3223
92	80.5580
93	67.1613
95	72.3789
96	78.2708
97	39.1237
98	84.6633
99	81.5354
101	77.2228
103	90.2136
105	91.8476
106	74.3868
107	85.0561
108	78.6019
110	74.0545
111	78.5946
113	89.3009
115	58.2348
116	73.6983
119	83.6050
120	66.5583
121	83.8636
122	85.1721
124	82.7016
125	80.8080

```
126         81.3095
129         86.0000
130         82.2538
131         73.1374
132         61.1894
134         78.8134
135         80.7926
137         80.1825
138         76.6993
139         68.4964
141         81.8333
142         82.5798
146         84.0308
154         73.9481
157         67.8662
164         77.2744
168         83.6250
170         65.8824
171         57.0468
173         75.7572
177         72.7684
180         58.1111
185         76.4162
216         68.4120
218         61.8211
221         70.7783
233         69.7811
234         70.7436
347         56.7320

Generating random graphs and computing average KNN...
```

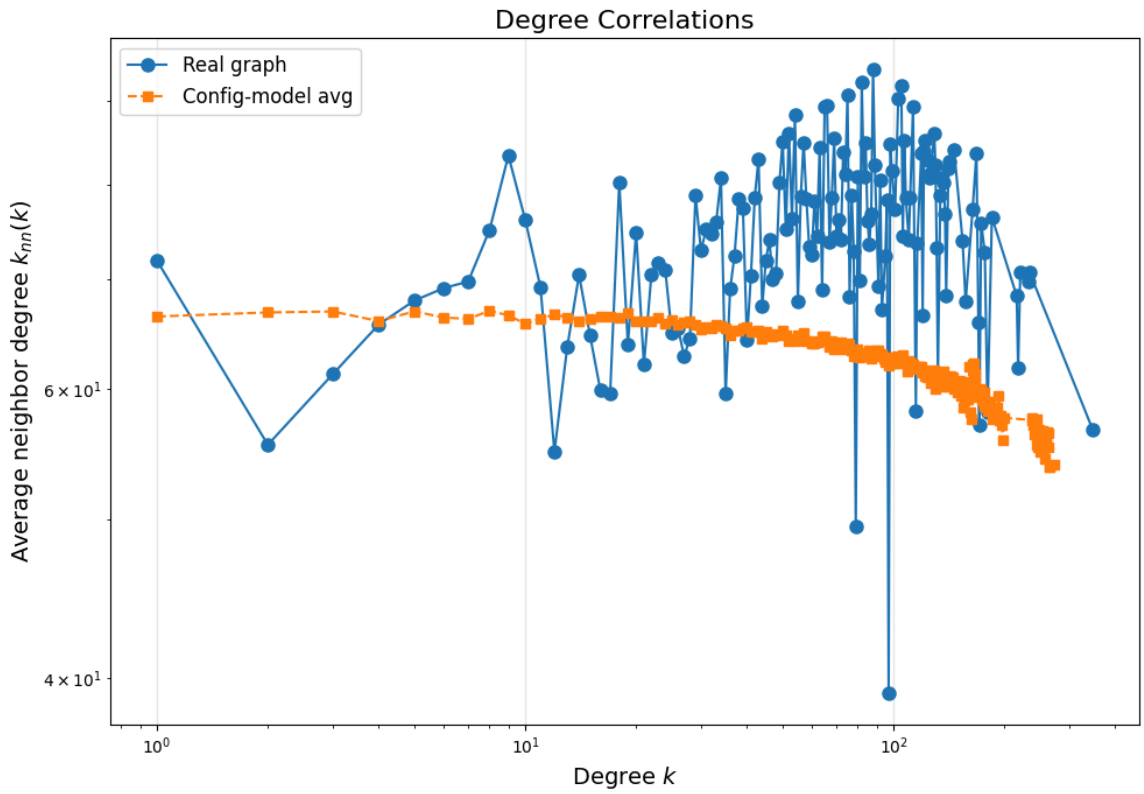
```
Random Graphs: 100%|██████████| 100/100 [00:02<00:00, 40.64it/s]
```

```
Configuration model average KNN vs k (100 instances):
Degree k   Average Neighbor Degree knn(k)
-----------------------------------
1          66.4553
2          66.8405
3          66.9300
4          66.0283
5          66.9389
6          66.3659
7          66.2283
8          67.0001
9          66.5621
10         65.7803
11         66.2555
12         66.6325
13         66.3885
14         66.0513
15         66.2106
16         66.4574
17         66.4346
18         66.3284
19         66.7504
20         65.9985
21         66.0349
22         66.0108
23         66.3414
24         65.8087
25         66.1543
26         65.6491
27         65.9575
28         66.0240
29         65.7356
30         65.2348
31         65.4461
32         65.3779
33         65.6840
34         65.6281
35         65.3372
36         64.7789
37         65.1111
38         65.1289
39         65.4255
40         65.5137
41         65.0848
42         65.0406
43         65.1210
44         64.4558
45         65.0393
46         64.8142
47         64.5758
48         64.7866
49         64.8893
50         65.1894
51         64.7381
52         64.2229
53         64.4153
54         64.2173
55         64.7398
56         64.2942
57         64.9254
```

58	64.1928
59	64.2071
60	64.0134
61	64.0082
62	64.1041
63	64.2225
64	64.5779
65	64.6086
66	64.1667
67	63.6950
68	64.2088
69	63.4961
70	63.7314
71	63.5514
72	64.1372
73	63.4429
74	63.5633
75	64.0290
76	63.8596
77	63.3180
78	62.8433
79	63.1933
80	63.4988
81	62.7302
82	63.0529
83	63.1811
84	62.8662
85	63.0374
86	63.3289
87	62.6253
88	63.1919
89	63.0505
90	63.3735
91	63.3244
92	62.8846
93	63.0748
94	62.6612
95	62.3593
96	62.5575
97	62.0596
98	62.2604
99	62.7091
100	62.4559
101	62.2162
102	62.8440
103	62.6623
104	62.2425
105	62.3133
106	62.9471
107	62.3608
108	62.2622
109	61.5599
110	62.4669
111	62.3526
112	62.0721
113	62.1405
114	61.9397
115	61.7909
116	61.7153
117	61.9338

118	61.9553
119	61.8905
120	61.4659
121	61.5859
122	61.0802
123	61.0428
124	61.3926
125	61.5350
126	60.4896
127	60.8590
128	61.1951
129	61.5798
130	60.0645
131	60.2450
132	61.0218
133	61.0022
134	60.4001
135	60.4676
136	60.2102
137	61.5378
138	60.4118
139	60.8206
140	60.5519
141	60.2431
142	60.8336
143	61.0889
144	60.7543
145	61.0396
146	60.0822
147	60.4690
148	60.1873
149	59.7015
150	60.0359
151	60.6772
152	59.4064
153	59.9227
154	60.1910
155	58.4504
156	59.1030
157	59.4157
158	59.9217
159	59.1975
160	60.6750
161	61.9627
162	58.1111
163	57.5072
164	62.1555
165	62.2409
166	61.3133
167	60.6443
168	59.3467
169	59.6906
170	59.8753
171	59.8408
172	60.0517
173	59.4655
174	59.0319
175	59.5845
176	59.7485
177	58.5301

178	58.6522
179	58.4839
180	58.2093
181	58.7715
182	58.4101
183	58.9645
184	58.3925
185	58.2822
186	57.5244
187	58.1961
188	58.3697
189	58.3630
190	58.3958
191	58.4969
192	57.8203
193	59.4214
194	57.4091
195	57.5164
196	57.7946
197	57.0355
198	55.8333
200	57.6400
237	57.4599
238	57.6282
239	57.0209
240	57.3667
241	56.3071
242	57.0231
244	55.4611
245	57.5388
246	55.2870
247	56.5128
248	56.7130
249	56.4418
250	54.9840
251	56.0876
252	56.5198
253	56.5968
254	55.0420
255	55.8941
256	56.5417
257	54.4397
258	56.1964
259	55.9498
260	56.0654
261	56.5307
263	56.4221
264	55.3371
265	53.8019
266	53.7594
273	53.9267




```
Assortativity coefficient for real network: -0.0110
Random Assortativity: 100%|██████████| 100/100 [00:05<00:00, 18.50it/s]
Average assortativity coefficient for random networks: -0.0447 ± 0.0063
```

ANALYSIS: The plot compares the degree correlation k_nn(k)—the average degree of neighbors of nodes with degree k—for the real-world email network (blue curve) and an ensemble of 100 randomized configuration models (orange curve).

Real Network: The real graph exhibits noticeable degree correlations, with a rising trend at lower degrees and more variability at higher degrees. This suggests assortative mixing for small-to-mid degree nodes (i.e., nodes tend to connect to similarly connected nodes), and complex structural patterns not captured by randomness.

Configuration Model: The randomized graphs generated by the configuration model show a flat or slightly decreasing trend, reflecting no significant degree correlation. This is expected, as the configuration model preserves only the degree sequence but randomizes the connections.

Conclusion: The contrast between the two curves indicates that the real-world email network has non-random structural properties, possibly due to community structure or functional organization, which are absent in the random models.