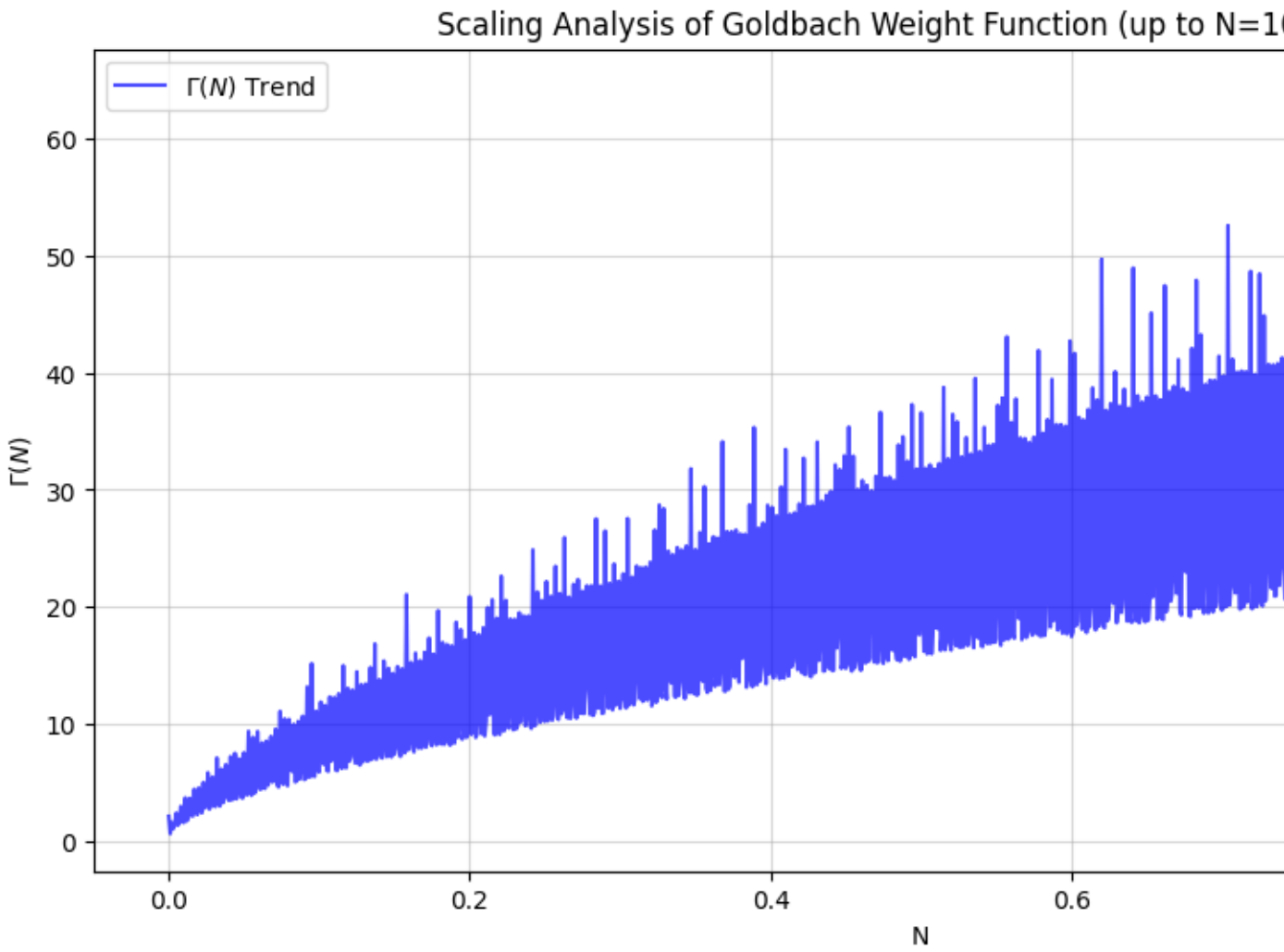
```
import math
import numpy as np
import matplotlib.pyplot as plt
from time import time

# Оптимизирано сито на Ератостен
def sieve(n):
    is_prime = np.ones(n+1, dtype=bool)
    is_prime[:2] = False
    for i in range(2, int(n**0.5) + 1):
        if is_prime[i]:
            is_prime[i*i:n+1:i] = False
    return np.where(is_prime)[0]

# Изчисляване на Gamma(N)
def gamma_goldbach(N, primes_list, primes_set):
    s = 0.0
    for p in primes_list:
        if p > N // 2:
            break
        q = N - p
        if q in primes_set:
            s += 1.0 / (math.log(p) * math.log(q))
    return s

# Параметри на теста
N_limit = 1000000  # Един милион
step = 1000        # През колко числа да записва точка за графиката

print(f"Стартиране на симулация до N = {N_limit}...")
```

```
start_time = time()

primes = sieve(N_limit)
primes_set = set(primes)

results_N = []
results_gamma = []

# Тестваме през определена стъпка, за да не претоварим паметта
for N in range(4, N_limit + 1, step):
    if N % 2 != 0: N += 1 # Само четни
    g = gamma_goldbach(N, primes, primes_set)
    results_N.append(N)
    results_gamma.append(g)

end_time = time()

print(f"ТЕСТЪТ ПРИКЛЮЧИ за {end_time - start_time:.2f} секунди.")

# Генериране на графика
plt.figure(figsize=(12, 6))
plt.plot(results_N, results_gamma, label=r'$\Gamma(N)$ Trend', color='blue', alpha=0.7)
plt.title(f"Scaling Analysis of Goldbach Weight Function (up to N={N_limit})")
plt.xlabel("N")
plt.ylabel(r"$\Gamma(N)$")
plt.grid(True, which="both", ls="-", alpha=0.5)
plt.legend()
plt.show()
```

```
Стартиране на симулация до N = 1000000...
ТЕСТЪТ ПРИКЛЮЧИ за 19.80 секунди.
```


Scaling Analysis of Goldbach Weight Function (up to N=1
Γ(N) Trend
60
50
40
30
20
10
0
Γ(N)
0.0
0.2
0.4
0.6
N