A Recursive Formula for Estimating Prime Number Density

VARUN MENON

Abstract: This paper introduces a method for estimating the proportion of prime numbers up to a given upper bound. It is based on a recursive approach inspired by the Sieve of Eratosthenes. This method works by taking the range of numbers given and iteratively removing the percentages of composite numbers that have specific prime factors. This approach is specific and flexible and can be applied to different ranges, providing a systematic way to estimate the prime density, similar to the Prime Number Theorem. A Python script is included to demonstrate test results using this method in comparison to actual values and values predicted by the Prime Number Theorem.

1. Introduction

Understanding the distribution of prime numbers is a fundamental problem in number theory, and it has significant implications in cryptography, computer science, and mathematics. Two methods referenced in this paper are the Sieve of Eratosthenes and the Prime Number Theorem. The Sieve of Eratosthenes works by identifying prime numbers by removing multiples of known primes. The Prime Number Theorem gives an asymptotic formula for the density of primes. This method is based on the Sieve of Eratosthenes, by iteratively removing proportions of numbers in the range that are multiples of known primes in order to output a final ratio that fairly accurately represents the proportion of primes in that range. In this paper, we present the underlying formulas, the logic behind the formulas, and then compare the test results with known values and predictions from the Prime Number Theorem.

2. Formulas and Definitions

1. Composite Numbers Estimation Formula ***C(x)*:**

$$C(x) = \sum_{n=1}^{k} \frac{1}{P_n}\left(1 - \sum_{i=1}^{n-1} C_i\right) - \frac{k}{x}$$

- ***C(x):*** The estimated proportion of composite numbers up to a given upper bound x.
- ***P_n**:* The nth prime number (e.g., 2, 3, 5, 7,...).
- ***k***: The maximum number of iterations, where primes $P_n \leq \sqrt{x}$.

- ***x***: The upper bound given.
- The inner summation represents the contribution of previous composite estimations C_i.
- The term $\frac{k}{x}$ accounts for the primes themselves used in the estimation.

2. Prime Numbers Estimation Formula ***P(x)*****:**

$$P(x) = 1 - C(x)$$

- ***P(x)***: The estimated proportion of prime numbers up to x.
- The sum of the proportion of prime numbers and composite numbers must equal 1.

3. Methodology

The formulas for estimating the density of prime numbers are based on the principle behind the Sieve of Eratosthenes, where composite numbers are systematically eliminated by iterating through primes. The key logic behind this approach is to:

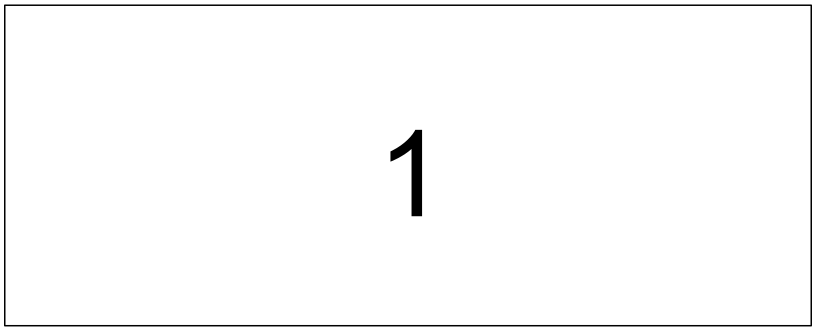
1. **Identify Composite Numbers:** Each iteration removes a percentage of numbers divisible by a specific prime, starting with the smallest prime (2) and moving upward. This process reflects the structure of the Sieve of Eratosthenes, where multiples of each prime are "marked" as composite.
2. **Recursive Subtraction of Contributions**: The formula for composite numbers ***C(x)*** accounts for the fact that a number may be divisible by multiple primes. The term $\sum_{i=1}^{n-1} C_i$ captures the accumulated contributions of smaller primes in each iteration, ensuring that composite numbers divisible by more than one prime are not overcounted.
3. **Prime Proportion Adjustment**: The adjustment term $\frac{k}{x}$ is then subtracted from ***C(x)*** to account for the P_n values that were not accounted for because they were used in the summation to determine composite numbers. After accounting for composite numbers, the formula for prime numbers ***P(x)*** subtracts the composite proportion from 1 (since a number is either prime or composite).
4. **Bound for Iterations** ($\sqrt{x}$): The process of removing composite numbers is conducted until the value of P_n exceeds $\sqrt{x}$. This is because any number larger than $\sqrt{x}$ that is composite must have a divisor smaller than $\sqrt{x}$. Therefore,

checking divisibility by primes up to $\sqrt{x}$ captures all composites within the range, making this an optimal stopping point for iterations. This limits the computational cost while maintaining accuracy. As more primes are included and the bound increases, the proportion of primes decreases logarithmically, which correlates with the Prime Number Theorem.

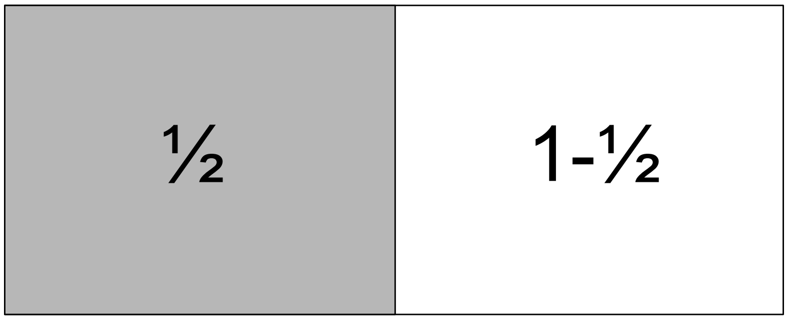
5. Multiply the proportion by the range x if looking for a number of primes from 0 to x instead of a ratio between 0 and 1.

4. Visual Representation

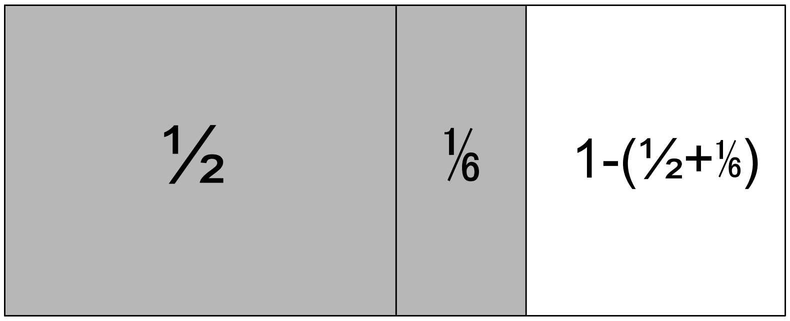
1. Let the white box below represent the proportion of numbers in a range from 0 to x.



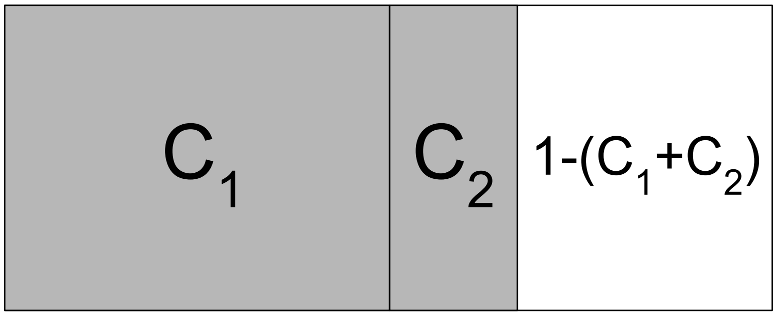
2. We will use shaded regions to represent each iteration of the recursive formula, representing composite proportions that we already accounted for. The first iteration tells us that half of all the numbers in the range are composite.



3. The shaded section represents the total proportion of the numbers we checked to be composite $\sum_{i=1}^{n-1} C_i$, and therefore the white section represents $1 - \sum_{i=1}^{n-1} C_i$.



4. We are only checking for primes that we have not already established as composite in previous iterations, so therefore we are only selecting primes from the "white box." The next prime after 2 is 3. One third of all the primes from the white box (which is ½ right now) must be composite, and since the white box accounts for $1 - \sum_{i=1}^{n-1} C_i$, we multiply $\frac{1}{P_n}$ by the current proportion of the "white box."



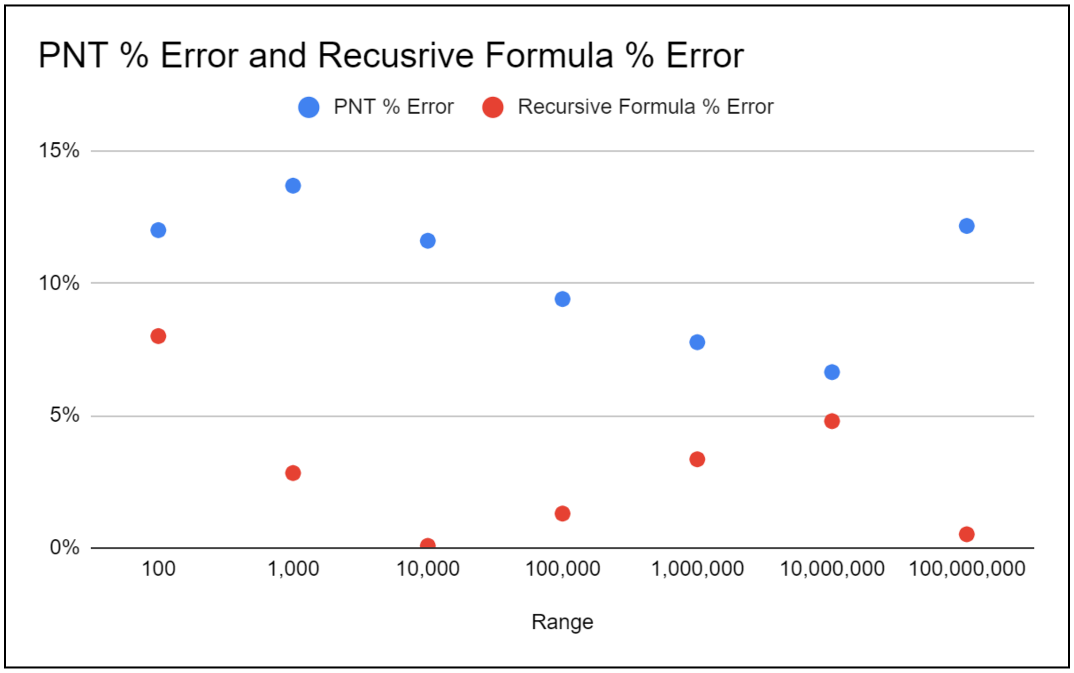
5. Summating all C_n until value of P_n exceeds $\sqrt{x}$ results in the total value of the shaded region. Then subtracting the total composite probability from 1 results in the value of the "white box."
6. Then the adjustment term $\frac{k}{x}$ is then subtracted to account for the P_n values that were not accounted for in the final result because they were used in the iterations themselves.

5. Testing and Results

In order to test the formula for accuracy across different number ranges, we implemented a script to run the formula for range values of 100, 1000, 10,000, 100,000, 1,000,000, 10,000,000, and 100,000,000. Then we compared the results with the actual prime number counts and estimates through the Prime Number Theorem (PNT), along with the percent error of both estimation methods for each trial.

The Python script used for testing the estimation method is located in the appendix.

Range	# of Primes	PNT	PNT % Error	Recursive Formula	Recursive Formula % Error
100	25	22	12%	27	8%
1,000	168	145	13.68%	164	2.83%
10,000	1229	1086	11.6%	1228	0.08%
100,000	9592	8691	9.4%	9717	1.3%
1,000,000	78498	72407	7.77%	81133	3.35%
10,000,000	664579	620421	6.64%	696459	4.79%
100,000,000	6058249	5429153	12.16%	6089698	0.519%



1. In all cases, the recursive formula was more accurate than the Prime Number Theorem, notably in the trial where the range was 10,000, the recursive formula only had a 0.08% error and was only off by 1 prime number.

2. This can be attributed to the nature of the Prime Number Theorem, where it basically says that the function becomes more accurate with higher ranges. Since we were dealing with relatively small ranges, the recursive formula worked better, as the iterative nature accounted for the primes themselves rather than the general nature of prime density.
3. The limitations of the recursive formula include computational complexity for large numbers, as each step requires extremely precise calculations for ideal results. The iterative nature also means that the method is computationally intensive when more iterations are required.

6. Appendix

A. Python Code for Prime Number Estimation using the method described in this paper

The following Python script was used to estimate the number of prime numbers up to a given upper bound *x* using the proposed recursive method.

```python
import math
from decimal import Decimal, getcontext

# Set precision for Decimal calculations
getcontext().prec = 20  # You can adjust this value for even higher precision


def is_prime(num):
    """Check if a number is prime."""
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True


def calculate_primes_ratio(x):
    """Calculate the ratio of prime numbers up to x using the specified method."""
    primes = []
    x = Decimal(x)  # Use Decimal for high precision
    limit = Decimal(math.sqrt(x)) + 1

    # Finding primes less than or equal to sqrt(x)
    for num in range(2, int(limit)):
        if is_prime(num):
            primes.append(Decimal(num))

    # Calculate composite contributions using the formula C(x)
    C = Decimal(0)  # Total composite contributions
    previous_C = Decimal(0)  # C_i for previous iterations

    for k in range(len(primes)):
        P_k = primes[k]
        # Calculate the current contribution to C based on previous contributions
        C += (Decimal(1) / P_k) * (Decimal(1) - previous_C)
        previous_C += (Decimal(1) / P_k) * (
            Decimal(1) - previous_C
        )  # Update previous C

    # Calculate the ratio of prime numbers P(x)
    k = Decimal(len(primes))  # Total number of primes found
    C -= k / x

    P_x = Decimal(1) - C

    return P_x


# Main function to prompt for input and calculate the ratio of primes
def main():
    x = int(input("Enter the upper bound (x): "))
    prime_ratio = calculate_primes_ratio(x)
    print(f"Estimated ratio of prime numbers from 1 to {x}: {prime_ratio}")
    print(f"Estimated number of prime numbers from 1 to {x}: {round(prime_ratio * x)}")


if __name__ == "__main__":
    main()
```

7. References

1. Hardy, G. H., and Wright, E. M. An Introduction to the Theory of Numbers. 6th ed., Oxford University Press, 2008.
2. Montgomery, H. L., and Vaughan, R. C. Multiplicative Number Theory I: Classical Theory. Cambridge University Press, 2007.
3. Rosen, Kenneth H. Elementary Number Theory and Its Applications. 3rd ed., Morgan Kaufmann, 2005.
4. Granville, A. "The Prime Number Theorem." In Prime Numbers: A Computational Perspective, 2000.
5. Bombieri, E. "The Riemann Hypothesis." Proceedings of the International Congress of Mathematicians, 2002.