Prime Sieving Based on Cyclical Frequencies

1. Introduction

The study of prime numbers has long played a central role in mathematics, not only for its theoretical significance but also for applications in fields such as cryptography, information theory, and computing. Over time, various methods have been developed for generating or verifying primes, including the sieve of Eratosthenes, probabilistic primality tests, and factorization algorithms.

This work introduces an alternative approach based on empirical observations of numerical patterns. It enables the detection of prime numbers greater than 3 using cyclical properties associated with numbers of the form $6n \pm 1$. The core idea lies in the emergence of **regular frequencies** within a sequence built entirely from simple arithmetic operations.

2. Theoretical Foundation

Every natural number greater than 1 can be classified according to its residue modulo 6. Only numbers of the form $6n \pm 1$ can be prime when $n > 0$, as other classes correspond to multiples of 2 or 3.

3. Generation of the Base Sequence

This alternating pattern of $+2, +4$ produces the exact sequence of $6n \pm 1$. Two frequencies are associated with each base number:

- $f_2 = 4n \pm 1$
- $f_1 = f_2 + 4n$

4. Frequency-Based Sieve

Each base number generates a cancellation pattern by alternating f_1 and f_2, eliminating composites starting from its square. This respects the modular structure and applies only within the $6n \pm 1$ domain.

5. Direct Primality Evaluation

A number N is evaluated by checking whether a combination of frequencies f_1, f_2 satisfies:

$$\frac{N - \text{offset}}{f_1 + f_2} \in \mathbb{Z}$$

Example: 77 with base 5 collides immediately:

$$\frac{77 - 7}{10} = 7 \Rightarrow \text{Composite}$$

6. Observed Optimization and Pattern Reuse

Cancellation patterns tend to repeat in structured blocks, especially when reaching products of primes. This allows reuse and modular design, suggesting computational efficiency.

7. Computational Validation (Methodological Note)

Python-based implementations have confirmed the accuracy of this model. Earlier optimized versions even validated numbers like $2^{127}-1$ efficiently. The current code prioritizes structure over performance, leaving room for community-driven improvements.

8. Discussion

Advantages:

- No need for factorization.
- Includes all primes greater than 3.
- Modular reuse of patterns.

Limitations:

- Does not treat 2 and 3.
- Limited to numbers of the form $6n \pm 1$.
- Lacks a full formal mathematical proof.

Potential: Useful for educational and research applications.

9. Conclusions

This model offers a structured and replicable alternative for analyzing primality. Its modular logic supports both theoretical understanding and possible optimization in future implementations.

10. Predictive Frequency Incidence Analysis

Besides step-by-step traversal, the model allows a predictive method to determine whether a frequency will reach a number directly. The general expression is:

$$\frac{N - \text{offset}}{f} \in \mathbb{Z}$$

with:

- $\text{offset}_1 = f_1 + i$
- $\text{offset}_2 = f_1 + f_2 + i$

Previously used formulas:

$$\frac{N - f_1 - i}{f}, \quad \frac{N - (f_1 + f_2) - i}{f}$$

are particular cases of this general offset-based analysis.

Appendix: Python Code Example

Prediction-Based Collision Check

```
def check_collision(N, f1, f2, index, max_steps=10):
    f = f1 + f2
    offset = f1 + index
    for step in range(max_steps):
        if (N - offset) % f == 0:
            return True  # Collision detected
        offset += f1 if step % 2 == 0 else f2
    return False
```

This code applies the predictive strategy described in Section 9.1 to verify whether a number N is reached by an alternating frequency pattern.