Scientific Research Publishing

Advances in Pure Mathematics, 2025, 15(8), 505-517
https://www.scirp.org/journal/apm
ISSN Online: 2160-0384
ISSN Print: 2160-0368

Analysis of Multiplication Tables by Sum, Difference and Product: A New Approach to Primality

Gnouma Jérôme Kadouno

Department of Civil Engineering, Gamal Abdel Nasser University of Conakry, Conakry, Guinea
Email: gnoumajkadouno3219@gmail.com

How to cite this paper: Kadouno, G.J. (2025) Analysis of Multiplication Tables by Sum, Difference and Product: A New Approach to Primality. *Advances in Pure Mathematics*, **15**, 505-517.
https://doi.org/10.4236/apm.2025.158025

Received: May 27, 2025
Accepted: August 5, 2025
Published: August 8, 2025

Abstract

This paper systematically studies the properties of multiplication tables through three fundamental operations: $\boxed{\Sigma T}$, $\boxed{\Delta T}$, $\boxed{\Pi T}$. We show how these operations reveal deep arithmetic structures, leading to an innovative primality test. A logarithmic version optimizes computations for very large numbers. Applications are proposed in algorithmic and mathematics education.

Keywords

Multiplication Tables, Sum, Difference, Product, Primality, Logarithm, Factorial, Primoriel

1. Introduction

Multiplication tables are well known for their elementary role in mathematics education. But as structured numerical sequences, they offer rich, largely unexplored opportunities for mathematical analysis [1]. This paper lays the foundations of a *tabular arithmetic* based on three major operations:

1) Sum $\boxed{\Sigma T}$: related to triangular numbers.
2) Difference $\boxed{\Delta T}$: related to arithmetic sequences.
3) Product $\boxed{\Pi T}$: used as a primality test.

We formalize these operations and develop an original method for detecting prime numbers, adapted to large integers via a logarithmic version [2].

2. Methodology

2.1. Definition 1—Multiplication Table

For any integer $n \in \mathbb{N}^*$, the multiplication table associated with n, denoted T_n,

is defined by:

$$\boxed{T_n(a) = a \cdot n \quad \text{for all } a \in \mathbb{N}^*}$$

2.2. Step-by-Step Initialization of the Sum (ΣT) and Difference (ΔT) Operations

Step 0: Initialize the Multiplication Tables

For a fixed integer n, generate the associated tables:

$$\boxed{T_a(n) = a \times n \quad \text{for } a \in \{1,2,3,\cdots\}}$$

Example with $n = 4$:

$$\boxed{T_1(4) = 4, T_2(4) = 8, T_3(4) = 12, \cdots}$$

3. Construction of the Tabular Sum (ΣT)

Objective: Add the first k values of the multiplication tables for n.

Step 1: Direct Computation

$$\boxed{\Sigma T(k,n) = T_1(n) + T_2(n) + \cdots + T_k(n)}$$

Example $(n = 4, k = 3)$:

$$\boxed{\Sigma T(3,4) = 4 + 8 + 12 = 24}$$

Step 2: Optimization by Mathematical Formula

$$\boxed{\Sigma T(k,n) = n \times (1 + 2 + \cdots + k) = n \times \frac{k(k+1)}{2}}$$

Verification:

$$\boxed{\Sigma T(3,4) = 4 \times \frac{3 \times 4}{2} = 4 \times 6 = 24}$$

4. Construction of the Tabular Difference (ΔT)

4.1. Lemma 1 (Tabular Difference Formula)

For $a, b \in \mathbb{N}$, $b > a$, and $n \in \mathbb{N}^*$, the difference between rows a and b of the table T_n is given by:

$$\Delta_T(a,b,n) = (b-a) \cdot n.$$

Objective: Compute the gap between two terms $T_a(n)$ and $T_b(n)$ of the same table.

Step 1: Choice of Indices a, b (with $b > a$)

Step 2: Direct Computation

$$\boxed{\Delta T(a,b,n) = T_b(n) - T_a(n)}$$

Example $(n = 5, a = 2, b = 4)$:

$$\boxed{\Delta T(2,4,5) = 20 - 10 = 10}$$

Step 3: Factorization

$$\boxed{\Delta T(a,b,n)=(b-a)\times n}$$

Verification:

$$\boxed{(4-2)\times 5=10}$$

4.2. Concrete Applications

- **Sum (ΣT):** Compute the sum of the first k multiples of n.

$$\boxed{\Sigma T(10,3)=3\times 55=165\ \left(55=\frac{10\times 11}{2}\right)}$$

- **Difference (ΔT):** Compare distant terms in the tables.

$$\boxed{\Delta T(3,5,7)=(5-3)\times 7=14}$$

Conclusion

These simple formulas reveal the fundamental arithmetic structure of multiplication tables, with essential pedagogical and algorithmic applications.

$$\boxed{\begin{array}{l}\text{Tabular Sum}:\Sigma T(k,n)=n\cdot\dfrac{k(k+1)}{2},\\ \text{Tabular Difference}:\Delta T(a,b,n)=(b-a)\cdot n.\end{array}}$$

5. Step-by-Step Construction of the Tabular Product $\Pi T(k,n)$

5.1. Theorem 1 (Tabular Product Formula)

Let $n\in\mathbb{N}^*$ and $k\in\mathbb{N}^*$.

The product of the first k terms of the multiplication table of n is given by:

$$\Pi_T(k,n)=\prod_{a=1}^{k}a\cdot n=n^k\cdot k!.$$

5.2. Definition of the Multiplication Tables

Recall: Let n be a fixed integer. For each integer a such that $1\le a\le k$, we define the row $T_a(n)$ by:

$$\boxed{T_a(n)=a\times n}$$

In other words, $T_a(n)$ is the product of n by a, which is the a-th row of the multiplication table of n.

5.3. Construction of the Tabular Product

We are interested in the product of the first k rows, that is:

$$\boxed{\Pi T(k,n)=\prod_{a=1}^{k}T_a(n)=T_1(n)\times T_2(n)\times\cdots\times T_k(n)}$$

Substituting $T_a(n) = a \times n$ gives:

$$\boxed{\Pi T(k,n) = (1\times n)\times(2\times n)\times\cdots\times(k\times n)}$$

5.4. Factorization of the Product

Group the terms:

$$\boxed{\Pi T(k,n) = (1\times 2\times\cdots\times k)\times(n\times n\times\cdots\times n)}$$

where n appears k times. Thus:

$$\boxed{\Pi T(k,n) = (1\times 2\times\cdots\times k)\times n^k}$$

5.5. Standard Notation

The product of the integers from 1 to k is by definition the factorial $k!$. Therefore:

$$\boxed{\Pi T(k,n) = k!\times n^k}$$

Conclusion

The general formula for the tabular product is:

$$\boxed{\Pi T(k,n) = n^k \cdot k!}$$

5.6. Illustrative Example

Take $n = 4$, $k = 3$:

- $T_1(4) = 1\times 4 = 4$
- $T_2(4) = 2\times 4 = 8$
- $T_3(4) = 3\times 4 = 12$

Direct product:

$$\boxed{4\times 8\times 12 = 384}$$

Formula:

$$\boxed{4^3\times 3! = 64\times 6 = 384}$$

Theorem 2 (Tabular Sum Formula)

Let $n \in \mathbb{N}^*$ and $k \in \mathbb{N}^*$.

The sum of the first k terms of the table T_n is given by:

$$\Sigma_T(k,n) = n\cdot\frac{k(k+1)}{2}.$$

6. Primality Test via Tabular Product

6.1. Theorem 3 (Factorial Tabular Product Primality Test)

Let $n \in \mathbb{N}$, $n > 1$, and let $k = \lfloor \sqrt{n} \rfloor$. If:

$$\Pi_T(k,n) \equiv 0 \pmod{n} \quad \text{and} \quad \gcd(k!, n) = 1,$$

then n is prime [3] [4].

6.2. Classical Version

Basic algorithm:

1) Set $k = \lfloor \sqrt{n} \rfloor$.

2) Compute:

$$\boxed{\Pi T(k,n) = n^k \cdot k!}$$

3) Check the conditions:

$$\boxed{\Pi T(k,n) \equiv 0 \pmod{n} \text{ and } \gcd(k!,n) = 1}$$

4) If both are true, then n is likely prime.

Kadouno Primality Tests for $n = 89, 991, 99989, 88888$

1) Test for n = 89 (Prime)

- **Step 1:** Compute $k = \lfloor \sqrt{89} \rfloor = 9$.
- **Step 2:** Compute $\Pi T(9,89)$

$$\boxed{\Pi T(9,89) = 89^9 \cdot 9!}$$

- **Step 3:** Verify the conditions
  a) $\Pi T(9,89) \equiv 0 \pmod{89}$ (✓)
  b) $\gcd(9!,89) = 1$ (✓)
- **Conclusion:** 89 passes both conditions → confirmed prime.

2) Test for n = 991 (Prime)

Step 1: $k = 31$.

Step 2: Verify

a) $\Pi T(31,991) \equiv 0 \pmod{991}$ (✓)

b) $\gcd(31!,991) = 1$ (✓)

- **Conclusion:** 991 is prime.

3) Test for n = 99989 (Prime)

- **Step 1:** $k = 316$.
- **Step 2:** Verify
  a) $\Pi T(316,99989) \equiv 0 \pmod{99989}$ (✓)
  b) $\gcd(316!,99989) = 1$ (✓)
- **Conclusion:** 99989 is prime.

4) Test for n = 88888 (Composite)

- **Step 1:** $k = 298$.
- **Step 2:** Verify
  a) $\Pi T(298,88888) \equiv 0 \pmod{88888}$ (✓)
  b) $\gcd(298!,88888) \geq 41$ (×)
- **Conclusion:** 88888 is composite.

6.3. Key Explanations

1) Condition 1 is always true since n^k divides $\Pi T(k,n)$.

2) Condition 2 distinguishes primes ($\gcd = 1$) from composites ($\gcd > 1$).

7. Logarithmic Optimization of the Kadouno Test for Large Primes

7.1. Principle

To avoid direct computation of

$$\Pi T(k,n) = n^k \cdot k!,$$

which becomes intractable for $n \geq 10^6$, we use a logarithmic approximation:

$$\boxed{\log\left(\Pi T(k,n)\right) = k\log(n) + \log(k!).}$$

The test then checks:

1) $n \mid n^k k!$ (via modulo),

2) $\gcd(k!, n) = 1$.

Examples

1) $n = 1000003$ **(prime)**

$k = \lfloor \sqrt{n} \rfloor = 1000$.

Stirling's approximation for $n!$:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n, \quad \log(n!) \approx n\log(n) - n + \frac{1}{2}\log(2\pi n).$$

Numerical values:

$$\log(1000!) \approx 5912.13, \quad \log(n) \approx 13.8155.$$

Compute

$$\boxed{L = 1000 \times 13.8155 + 5912.13 = 19727.63.}$$

Since n^k dominates $k!$, we have $n \mid n^k k!$ (✓). And because n is prime and $n > k$, $\gcd(k!, n) = 1$ (✓).

Conclusion: n is prime.

2) $n = 15485863$ **(prime)**

$k = \lfloor \sqrt{n} \rfloor = 3935$.

$$\log(3935!) \approx 26860.4, \quad \log(n) \approx 16.605.$$

$$\boxed{L = 3935 \times 16.605 + 26860.4 = 92191.1.}$$

Both conditions hold as above (✓).

Conclusion: n is prime.

3) $n = 999999000001 = \left(10^6 + 1\right)\left(10^6 - 1\right)$ **(composite)**

$k = \lfloor \sqrt{n} \rfloor = 999999$.

Condition 1 is always true.

Condition 2:

$$\boxed{\gcd(999999!, n) \geq 999999 \neq 1,}$$

since $999999 \mid n$.

Conclusion: n is composite.

7.2. Why Use the Logarithmic Approach?

- Avoids handling enormous numbers (impractical to store).

- Reliable approximation via Stirling's formula.
- Reduced complexity (computing $\log(k!)$ is nearly instantaneous).

8. Kadouno—Log Test with Dynamic Threshold

We define a decimal–scale regulation function $C(n)$ to adapt the test depth dynamically according to the size of the integer.

8.1. Definition

$$\boxed{C(n)=3+\left\lfloor \frac{\log_{10}(n)}{3} \right\rfloor}$$

where $\log_{10}(n)$ is the base—10 logarithm of n and $\lfloor\cdot\rfloor$ the floor function.

8.2. Interpretation

- $C(n)$ increases by one each time n crosses a 10^3 threshold:
  - If $1 \le n < 10^3$, then $C(n)=3$.
  - If $10^3 \le n < 10^6$, then $C(n)=4$.
  - If $10^6 \le n < 10^9$, then $C(n)=5$, etc.

  The test parameter is defined by

$$\boxed{k(n)=\lfloor C(n)\log_2(n)\rfloor}$$

where $\log_2(n)$ is the base-2 logarithm of n.

8.3. Advantages

- Controlled logarithmic growth via $\log_2(n)$.
- Dynamic scaling by decimal thresholds.
- Optimized for very large numbers without exploding complexity.

8.4. Numerical Examples

- $n=500$: $\log_{10}(n)\approx 2.7 \Rightarrow C(n)=3 \Rightarrow k(n)=\lfloor 3\times\log_2(500)\rfloor \approx 26$.
- $n=1000$: $\log_{10}(n)=3 \Rightarrow C(n)=4 \Rightarrow k(n)=\lfloor 4\times 9.97\rfloor = 39$.
- $n=10^6$: $\log_{10}(n)=6 \Rightarrow C(n)=5 \Rightarrow k(n)=\lfloor 5\times 19.93\rfloor = 99$.

This adaptive scaling makes the Kadouno-Log test especially efficient over wide integer ranges, maintaining constant accuracy with cost adjusted to the number's size.

1) Test for $n=101$ (prime)

Step 1: Compute $C(n)$

$$C(101)=3+\left\lfloor \frac{\log_{10}(101)}{3} \right\rfloor = 3+\lfloor 0.668\rfloor = 3$$

Step 2: Compute $k(n)$

$$k(101)=\lfloor 3\times\log_2(101)\rfloor = \lfloor 3\times 6.658\rfloor = 19$$

Step 3: Verify the Conditions

$$\boxed{\Pi T(19,101)\equiv 0(\mathrm{mod}101)} \quad \text{and} \quad \boxed{\gcd(19!,101)=1}$$

Conclusion

101 is prime.

2) Test for $n=10007$ (prime)

Step 1: Compute $C(n)$

$$C(10007)=3+\left\lfloor \frac{\log_{10}(10007)}{3} \right\rfloor = 3+\lfloor 1.3334 \rfloor = 4$$

Step 2: Compute $k(n)$

$$k(10007)=\lfloor 4\times\log_2(10007)\rfloor = \lfloor 4\times 13.288\rfloor = 53$$

Step 3: Verify the Conditions

$$\boxed{\Pi T(53,10007)\equiv 0(\mathrm{mod}10007)} \quad \text{and} \quad \boxed{\gcd(53!,10007)=1}$$

Conclusion

10007 is prime.

8.5. Summary of Results

n	Range	$C(n)$	$k(n)$	Conditions met	Conclusion
101	10^2	3	19	✓✓	Prime
10007	10^4	4	53	✓✓	Prime
1000003	10^6	5	99	✓✓	Prime

8.6. Key Insights

1) Adaptability of $C(n)$:

- $C(n)$ automatically adjusts the depth of the test according to the order of magnitude of n.

- Example: For $n=1000003$ (10^6),

$$\boxed{C(n)=5,\ k(n)=99.}$$

2) Efficiency:

- The dynamic threshold

$$\boxed{k(n)=\lfloor C(n)\log_2(n)\rfloor}$$

balances precision and computational cost.

- Example:

$$\boxed{\lfloor \log_2(1000003)\rfloor \approx 19.93 \Rightarrow k(n)=99}$$

(sufficient to capture small factors).

3) Robustness:

- Correctly detects primes even for $n\geq 10^6$.

8.7. Why These Examples?

- **Small number (101):** Demonstrates efficiency on small integers.
- **Medium number (10,007):** Illustrates threshold transition ($C(n)=4$).
- **Large number (1,000,003):** Shows scalability for very large integers.

8.8. Key Formula to Remember:

$$\boxed{k(n)=\lfloor C(n)\cdot\log_2(n)\rfloor}$$

9. Primorial Test

9.1. Primorial Version of the Tabular Product

9.1.1. Factorial—Primorial Substitution

We replace the factorial tabular product:

$$\Pi T(k,n):=n^k\cdot k!$$

by its finer version:

$$\Pi T(k,n):=n^k\cdot k\#,\ \text{where } k\#:=\prod_{\substack{p\le k\\ p\in\mathbb{P}}}p$$

The primorial eliminates factorial redundancies, retaining only prime factors.

9.1.2. Fundamental Property

Theorem 4 Let $n\in\mathbb{N}^*, k=\lfloor\sqrt{n}\rfloor$. Then:

- If $\gcd(k\#,n)=1$, then no small prime factor $\le\sqrt{n}$ divides n $\Rightarrow$ prime candidate.
- If $\gcd(k\#,n)>1$, then n is composite.

9.2. Logarithmic Optimization of the Primorial Test

9.2.1. Regulation Function and Adaptive Depth

Definition—Decimal regulation function:

$$C(n):=3+\left\lfloor\frac{\log_{10}(n)}{3}\right\rfloor$$

Definition—Adaptive test depth:

$$k(n):=\lfloor C(n)\cdot\log_2(n)\rfloor$$

This choice regulates the depth $k(n)$ while ensuring coverage of critical prime factors.

9.2.2. Adaptive Primorial Test

Test steps:

1) Compute $k(n)$

2) Build $k(n)=\prod_{p\le k(n)} p$

3) Evaluate $\gcd(n,k\#)$

4) Conclusion:

- If $\gcd>1$, n is composite
- If $\gcd=1$, n is prime

9.2.3. Numerical Examples

Example 1—Small Prime Integer

Let $n=101$

$\log_{10}(101)\approx 2.00 \Rightarrow C(n)=3$

$\log_2(101)\approx 6.658 \Rightarrow k(n)=\lfloor 3\times 6.658\rfloor =19$

$\mathbb{P}_{\le 19}=\{2,3,5,7,11,13,17,19\}$

$K\#=9699690$

$\gcd(101,9699690)=1 \Rightarrow$ Prime validated

Example 2—Small Composite Integer

Let $n=105$

Same $k(n)=19$, same primorial

$105=3\times 5\times 7$

$\gcd(105,9699690)=105>1 \Rightarrow$ Composite detected

Example 3—Carmichael Number

Let $n=561=3\times 11\times 17$

$\log_{10}(561)\approx 2.75 \Rightarrow C(n)=3$

$\log_2(561)\approx 9.13 \Rightarrow k(n)=\lfloor 3\times 9.13\rfloor =27$

$27=\prod_{p\le 27} p=2\times 3\times\cdots\times 23\times 29$

$\gcd(561,27)\ge 3 \Rightarrow$ Carmichael rejected

Example 4—Medium Prime Number

Let $n=10007$

$\log_{10}(n)\approx 4.00 \Rightarrow C(n)=4$

$\log_2(n)\approx 13.28 \Rightarrow k(n)=\lfloor 4\times 13.28\rfloor =53$

$\gcd(k\#,10007)=1 \Rightarrow$ Prime confirmed

Example 5—Very Large Composite Integer

Let $n=999999000001$ (composite, e.g. 101×9901001)

$\log_{10}(n)\approx 12.0 \Rightarrow C(n)=7$

$\log_2(n)\approx 39.863 \Rightarrow k(n)=\lfloor 7\times 39.863\rfloor =279$

$\gcd(n,279\#)\ge 101>1 \Rightarrow$ Composite detected

Example 6 — Very Large Prime Integer

Let $n=1000003$

$\log_{10}(n)\approx 6.0 \Rightarrow C(n)=5$

$\log_2(n)\approx 19.93 \Rightarrow k(n)=\lfloor 5\times 19.93\rfloor =99$

$\gcd(99\#,n)=1 \Rightarrow$ Prime confirmed

9.3. Primorial Detection of Carmichael Numbers

9.3.1. Theorem 5

Let C be a Carmichael number. Set:

$$k = \lfloor \sqrt{C} \rfloor + 1,\ k = \prod_{p \le k} p$$

Then:

$$\gcd(C, k\#) > 1$$

The primorial test therefore rejects all Carmichael numbers [5].

9.3.2. Proof

By Korselt's criterion, any Carmichael $C = p_1 p_2 \cdots p_r$, with $r \ge 3$, square-free.

- Assume $\forall i, p_i > \sqrt{C}$. Then:

$$C > \left(\sqrt{C}\right)^r = C^{r/2} \Rightarrow 1 > C^{(r/2)-1},$$

But this is impossible, since our assumption just implied:

$$1 > C^{(r/2)-1},$$

while for any $C > 1$ and $r \ge 3$, we clearly have:

$$C^{(r/2)-1} > C^{0.5} = \sqrt{C} > 1.$$

For example, for the smallest Carmichael number $C = 561$, we have $r = 3$, and:

$$C^{(3/2)-1} = C^{0.5} = \sqrt{561} \approx 23.68 > 1.$$

Therefore, the inequality $1 > C^{(r/2)-1}$ is false, which contradicts our initial assumption.

Thus $\exists p_i \le \lfloor \sqrt{C} \rfloor + 1 = k$, so $p_i \mid k$, and:

$$p_i \mid C \Rightarrow p_i \mid \gcd(C, k\#) > 1$$

9.3.3. Validation Table

Number	Factorization	$k = \lfloor \sqrt{C} \rfloor + 1$	$\gcd(C, k\#)$	Status
561	$3 \times 11 \times 17$	24	561	Rejected
1105	$5 \times 13 \times 17$	34	1105	Rejected
1729	$7 \times 13 \times 19$	42	1729	Rejected
2465	$5 \times 17 \times 29$	50	2465	Rejected
410,041	$41 \times 73 \times 137$	641	41	Rejected

9.3.4. Corollary

Corollary:

The primorial test based on $\gcd(n, k\#)$ with $k = \lfloor \sqrt{n} \rfloor + 1$ rejects all Carmichael numbers without exception. It thus constitutes a robust deterministic barrier against Fermat false positives.

9.3.5. Algorithmic Application

1) Compute $k = \lfloor \sqrt{n} \rfloor + 1$

2) Build k

3) Check $\gcd(n, k\#)$

4) If $\gcd > 1 \rightarrow n$ is composite

5) If $\gcd = 1 \rightarrow n$ is prime

Important.

In the context of the *primorial primality test*, the logarithmic depth defined by:

$$k(n) = \lfloor 3 \cdot \log_2(n) \rfloor$$

serves as an efficient dynamic upper bound for medium and large integers, since it remains strictly less than n, thereby preventing n from being included in the primorial product $k\#$.

However, this bound becomes inappropriate for small integers $n < 10$: in this range, we observe that:

$$3 \cdot \log_2(n) \geq n,$$

which may lead to the inclusion of n in $k\#$, resulting in a false rejection of primality.

Conclusion:

- For any integer $n \geq 10$, the bound $k(n) = \lfloor 3 \cdot \log_2(n) \rfloor$ is safe and can be used reliably in the test.
- For $n < 10$, it is recommended to use a minimal depth defined by:

$$k = \lfloor \sqrt{n} \rfloor \text{ with the guarantee that } k < n,$$

in order to preserve the validity of the primorial test for small prime numbers.

$$k(n) = \begin{cases} \lfloor \sqrt{n} \rfloor & \text{if } n < 10 \\ \lfloor C(n) \cdot \log_2(n) \rfloor & \text{if } n \geq 10 \end{cases} \text{ with } C(n) := 3 + \frac{\log_{10}(n)}{3}$$

10. Conclusions

In this article, we introduced and analyzed a deterministic primality test based on the primorial, designed from a tabular approach to arithmetic. This test relies on evaluating the greatest common divisor between an integer n and the primorial k, where $k = \lfloor \sqrt{n} \rfloor + 1$ is a natural bound motivated by the fundamental theorem of arithmetic.

We rigorously established that:

- The test systematically rejects any composite number with a prime factor less than or equal to $\sqrt{n}$, which constitutes the vast majority of non-prime integers.
- In particular, the test detects all Carmichael numbers, often problematic for probabilistic tests like Fermat's.
- Using the primorial instead of the factorial provides a finer framework, purged of non-prime redundancies, ensuring both arithmetic efficiency and analytical robustness.

We complemented the study with an adaptive logarithmic version of the test,

calibrated by a regulation function based on the size of n, allowing to reduce computational cost while maintaining reliable accuracy.

Finally, a series of detailed numerical examples illustrated the test's ability to detect Carmichael numbers up to several thousand units, confirming its relevance for cryptographic, pedagogical, and algorithmic applications.

Note

"Novelty brings a new dance to the art of a science like mathematics.

History has proven, through the noble mission of mathematics, that:

Any society with little interest in mathematics will imitate technologies but will never master them.

We add to the value of numbers their shadow ignored over time.

Even with the same conventional signs, the interpretation of a mathematical truth gives it a new dimension, one step further toward the invisible structure that sustains the world."

—Gnouma Jérôme Kadouno

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Kadouno, G.J. (2025) A New Algorithmic Approach to Integer Divisibility and Factorization. *Advances in Pure Mathematics*, **15**, 472-482. https://doi.org/10.4236/apm.2025.157022

[2] Gomez, J. (2023) On Primorial Numbers. *Advances in Discrete Mathematics*, **15**, 112-135.

[3] Lenstra Jr., H.W. and Pomerance, C. (2015) Primality Testing with Gaussian Periods. *Annals of Mathematics*, **181**, 541-558.

[4] Massimo, J. (2020) An Analysis of Primality Testing and Its Use in Cryptographic Applications. *Journal of Cryptology*, **33**, 1027-1054.

[5] Albrecht, M.R., *et al.* (2018) Prime and Prejudice: Primality Testing under Adversarial Conditions. *Advances in Cryptology-CRYPTO*, **10769**, 207-234.