Reframing Prime Theory: A Fresh and Computer-aided Perspective

Author: Philip Pan

An introduction

Traditional research in number theory primarily revolves around the Eratosthenes sieve method and other methodologies, resulting in significant achievements. Despite the application of numerous complex mathematical techniques, certain challenging issues persist. For example, Goldbach’s Conjecture, has yet to be proven. It is therefore reasonable to assume that a fresh, innovative approach is required, and strong foresight is crucial.

It is out of this consideration that new definitions are made as follows:

1. The integer 1 is the first and the smallest prime number, leaving the integer 2 out；

2. All prime numbers are odd；

3. The set of odd numbers possess two subsets: (1) the set of prime numbers and (2) the set of odd composite numbers;

4. The set of odd composite numbers can be expressed by the products of two or more prime numbers;

5. Therefore, the set of odd numbers minus the set of the products of two or more prime numbers equals the set of prime numbers.

Thus, a formula for prime numbers is proposed:

1. The formula for odd numbers is So = 2n + 1, where n = 1, 2, 3, ..., n....

2. The formula for odd composite numbers, denoted as Sc, follows a sequence of prime products:

   Sc1= 3*3, 3*5, 3*7, ... 3*(3+2k) where k=0, 1, 2, 3, ...;
   Sc2= 5*5, 5*7, 5*9, ... 5*(5+2k) where k=0, 1, 2, 3, ...;
   Sc3=7*7, 7*9, 7*11, ...7*(7+2k) where k=0, 1, 2, 3, ...; ......
   ......
   Scx =x*x, x*(x+2), x*((x+2)+2), ... x*(x+2k) where k=0, 1, 2, 3, ...

   Since the number 1 does not make any differences in calculation of products, it is not included for convenience.

   Based on this sequence, a python code is programmed as follows:

```
def generate_sequence(x, n_terms):
    """Generates the sequence Sc_x = x*x, x*(x+2), x*(x+4), ..., for n_terms"""
    sequence = []
    for k in range(n_terms):
```

```
            term = x * (x + 2 * k)
            sequence.append(term)
        return sequence

    def generate_all_sequences(start, n_sequences, n_terms):
        """Generates all sequences from Sc_start to Sc_(start + n_sequences)"""
        all_sequences = {}
        for i in range(n_sequences):
            current_x = start + 2 * i
            all_sequences[f'Sc_{i+1} (x={current_x})'] = generate_sequence(current_x,      n_terms)
        return all_sequences

    # Example: Generate 5 sequences, each with 6 items
    n_sequences = 10
    n_terms = 10
    start_value = 3    # Starting value for the sequences

    sequences = generate_all_sequences(start_value, n_sequences, n_terms)

    # Display sequences
    for key, seq in sequences.items():
        print(f"{key}: {seq}")

    Sc_1 (x=3): [9, 15, 21, 27, 33, 39, 45, 51, 57, 63]
    Sc_2 (x=5): [25, 35, 45, 55, 65, 75, 85, 95, 105, 115]
    Sc_3 (x=7): [49, 63, 77, 91, 105, 119, 133, 147, 161, 175]
    Sc_4 (x=9): [81, 99, 117, 135, 153, 171, 189, 207, 225, 243]
    Sc_5 (x=11): [121, 143, 165, 187, 209, 231, 253, 275, 297, 319]
    Sc_6 (x=13): [169, 195, 221, 247, 273, 299, 325, 351, 377, 403]
    Sc_7 (x=15): [225, 255, 285, 315, 345, 375, 405, 435, 465, 495]
    Sc_8 (x=17): [289, 323, 357, 391, 425, 459, 493, 527, 561, 595]
    Sc_9 (x=19): [361, 399, 437, 475, 513, 551, 589, 627, 665, 703]
    Sc_10 (x=21): [441, 483, 525, 567, 609, 651, 693, 735, 777, 819]
```

3. The formula for prime numbers, denoted as Sp, can be defined as the result of subtracting Sc from So:

   Sp = So - Sc.

   According to this formula, a python code is programmed as follows:

```
defgenerate_sequence_so(limit):
    # Generate sequence So: So_n = 2n + 1 for n = 1, 2, 3, ...
    So = []
    n = 1
    while True:
        term = 2 * n + 1
        if term > limit:
            break
```

```
        So.append(term)
        n += 1
    return So

def generate_sequence_sc(limit):
    # Generate sequence Sc with multiple sub-sequences
    Sc = set() # Use a set to ensure unique terms in Sc
    for x in range(3, limit, 2): # x = 3, 5, 7, ...
        k = 0
        while True:
            term = x * (x + 2 * k)
            if term > limit:
                break
            Sc.add(term) # Only unique terms are added
            k += 1
    return sorted(Sc)

def find_common_terms(So, Sc):
    # Find terms that are common in So and Sc
    common_terms = [term for term in So if term in Sc]
    return common_terms

def subtract_sequences(So, Sc):
    # Perform So - Sc only on common terms
    common_terms = find_common_terms(So, Sc)
    result = [term for term in So if term not in common_terms]
    return result

# Define a limit for terms in sequences
limit = 2000

# Generate sequences So and Sc
sequence_so = generate_sequence_so(limit)
sequence_sc = generate_sequence_sc(limit)

# Perform So - Sc according to the given conditions
result = subtract_sequences(sequence_so, sequence_sc)

# Output the results
# print("Sequence So:", sequence_so)
# print("Sequence Sc:", sequence_sc)
# print("Common terms (So ∩ Sc):", find_common_terms(sequence_so, sequence_sc))
print("Result of So - Sc:", result)
```

Result of So - Sc:

[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97,
101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199,

211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293,
307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397,
401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499,
503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599,
601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691,
701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797,
809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887,
907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997,
1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097,
1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193,
1201, 1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297,
1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381, 1399,
1409, 1423, 1427, 1429, 1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499,
1511, 1523, 1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597,
1601, 1607, 1609, 1613, 1619, 1621, 1627, 1637, 1657, 1663, 1667, 1669, 1693, 1697, 1699,
1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783, 1787, 1789,
1801, 1811, 1823, 1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889,
1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997, 1999]
(hand-edited)

It is worth attention that the subtraction (So-Sc) is not executed unless the following two conditions are fully satisfied:

(1) Within sequence Sc, if any one term or any terms are equal to one another, only one of those terms is chosen for the subtraction (So-Sc), in order to prevent any duplication;
(2) Only when one term in sequence Sc exactly matches one term in sequence So;

Discussions

1. The exclusion of the number 2 from the set of prime numbers may spark debates. Since the inception of prime number theory, many mathematicians have argued for the inclusion of 2 as a prime. However, it can be argued that numbers, particularly prime numbers, are not totally subject to human thinking but are instead objective entities. In this context, applying the methodology of physics—grounded in observation, verification, revision, and theoretical development—might be more relevant, although rigorous reasoning and logic remain crucial.

   It can also be argued that, in many number theories, the number 2 is often left outside of mathematicians' primary focus. This is likely because it is the only even number, which makes it unique compared to other primes.

2. The next question is:

   Does Sc, the sequence of products of primes, encompass every element of the set of odd composite numbers?

   The Fundamental Theorem of Arithmetic[1] states that every integer greater than 1 is either a prime number or can be uniquely factored into prime numbers, and this means that any integer greater than

1 can be expressed as a product of prime numbers, e.g., 3=1*3, 15=3*5, and 35=5*7.

Now, we suppose there is a product of two randomly chosen odd numbers, pr = m × n (m ≠ n), and because they are randomly picked up, they are capable of expressing any elements inside the set of odd composite numbers. Four possible outcomes logically follow:

(1) the product is an odd number (due to basic arithmetic properties);
(2) the product is a prime when either m or n is the number 1;
(3) either m or n is a product of primes while the other is a prime;
(4) both m and n are primes greater than 1, or they are products of primes.

Except outcome (2), all the others are elements of the set of odd composite numbers. That is,

pr ⊂ Sc

If any of these products lies outside the set of odd composite numbers, it would contradict the Fundamental Theorem of Arithmetic.

Thus, we conclude that any element in the set of odd composite numbers can indeed be expressed as a product of prime numbers.

3. By increasing the upper limit in the python code mentioned above, the values of primes can be generated as large as the computer's processing capability allows.

4. An implication for Goldbach's Conjecture

According to the formula Sp = So - Sc, So is always odd numbers while its elements equivalent to Sc are subtracted by Sc, leaving Sp odd.

Goldbach's Conjecture can be reformulated as follows:

> For any two prime numbers—whether they are as close as twin primes or separated by a large gap—their sum is always an even number.

Conclusions

The study of primes should adopt a new, innovative approach. Excluding the prime number 2, a formula for generating primes can be established: the set of odd numbers minus the set of odd composite numbers yields the set of primes. From this formula, it is evident that primes, aside from 2, must always be odd. Consequently, when two odd prime numbers are added, their sum is invariably even.

Reference

1. Hardy, G. H. & Wright, E. M. (2019) An Introduction to the Theory of Numbers, p3.