Matlab Experiment 5 Report: Prime Numbers

Yikun Zhang[*]
SID: 14336275
Major: Mathematics and Applied Mathematics

Abstract

As the most mysterious part of mathematics, number theory attracts mathematicians' interest by just some simple integer, prime numbers. In this experiment we are supposed to figure out some algorithms which can generate prime numbers. Furthermore, we are also interested in formulating some expressions that can yield infinitely many prime numbers. At last, the distribution of prime numbers on the real line also enthralls us to carry out computer programming to explore them and approach those famous open problems.

Contents

[*]School of Mathematics, Sun Yat-sen University

1 Introduction and Purpose

Within the realm of Mathematics, number theory is one of the most mysterious fields. It is rich in problems of an especially vexing sort: they are tantalizingly simple to state but notoriously difficult to solve. Among all those attractive concepts, prime numbers served as the foundation of number theory.

By definition, a **prime number** (or a prime) is a natural number greater than 1 that has no positive divisors other 1 and itself. A natural number greater than 1 that is not a prime number is called a composite number.[1] It turns out, as demonstrated by Euclid around 300 BC, that there are infinitely many prime numbers. The reason lies in the fact that if there are only finitely many prime numbers, we can simply multiply these number together and add 1 to the resulting product, yielding a new number which cannot be divided by the "existing" prime number. Such an elegant proof via contradictory arguments encourage many mathematicians to conjecture up some millennium problems, such as the so-called Goldbach conjecture (that every even integer greater than 2 can be expressed as the sum of two primes), and twin prime conjecture (that there are infinitely many pairs of primes whose difference is 2)[1].

In this experiment, we aim to develop some algorithms to judge whether a given integer is prime. Moreover, we are interested in investigating the infiniteness of prime numbers. At this stage, a well-known category of prime number, Mersenne numbers, is also scrutinized. In addition, we are eager to figure out a polynomial function with integer coefficients that always take prime numbers as its output. Unfortunately, it is impossible and an outline of a proof will be proposed. Last but not least, the distribution of prime numbers on the real line will also be visualized. All these experiments is based on computer programming on the platform of Matlab 2016a, which may approximately give us the overview of these problems. Essentially, the main theme of our experiments will follow the Chapter 5 of the book "Mathematical Experiments" written by Shangzhi Li et.al[2].

2 Methods and Results

2.1 How can we figure out prime number?

To find out all the prime numbers that is less than or equal to a given positive integer, an ancient Greek scholar called Eratosthenes proposed a stepwise method to eliminate the composite numbers from our desired prime number set. The procedures are rather intuitive: we first eliminate all the multiples of 2 except 2 itself, all the multiples of 3 except 3 itself, then 5, 7, etc. With a preassigned stoping criterion, the resulting set of numbers is consisted of all the prime numbers up to any given limit, says N. This is the so-called sieve of Eratosthenes.

When we implement this algorithm by setting the stopping criterion as not other integers in the set can be divided by the determined prime numbers in the preceding steps. We encapsulated the sieve algorithm into a function called *"PRIME"*. See the codes in the Appendix. Other stopping criteria like the looping times are greater than $\sqrt{N}$ are also capable in this algorithm.

In reality, the sieve of Eratosthenes is utilizing multiplications to search for prime numbers. On the contrary, we can make use of divisions to justify whether an integer is prime. Suppose that we have already figured out n prime numbers. To obtain the next prime number, we only have to check whether $p_n + 2$ can be divided by its previous prime numbers $p_1 = 2, p_2 = 3, ..., p_n$. If it does, then $p_n + 2$ is composite. Otherwise, it is another prime number.

To find out all prime numbers less than $N = 10000$, we compare the efficiency of the sieve of Eratosthenes and the tentative division method via Matlab programming. Contrary to the claim in the textbook[2], the sieve of Eratosthenes spends 0.0189 second while the tentative division method spends 0.3025 second. The reason might be that we choose an usual stopping criterion and use vector-form programming skills in the implementation of the sieve of Eratosthenes.

However, it is unrealistic to obtain some enormous prime number using either the sieve of Eratosthenes or the tentative division method, since the computational process of these algorithms is relatively slow. Thus we need to resort to other methods so as to generate large prime numbers.

Now we are interested in the remainder of 2^{n-1} divided by n, where $n = 2, 3, ..., 100$. See Figure 1 for details.

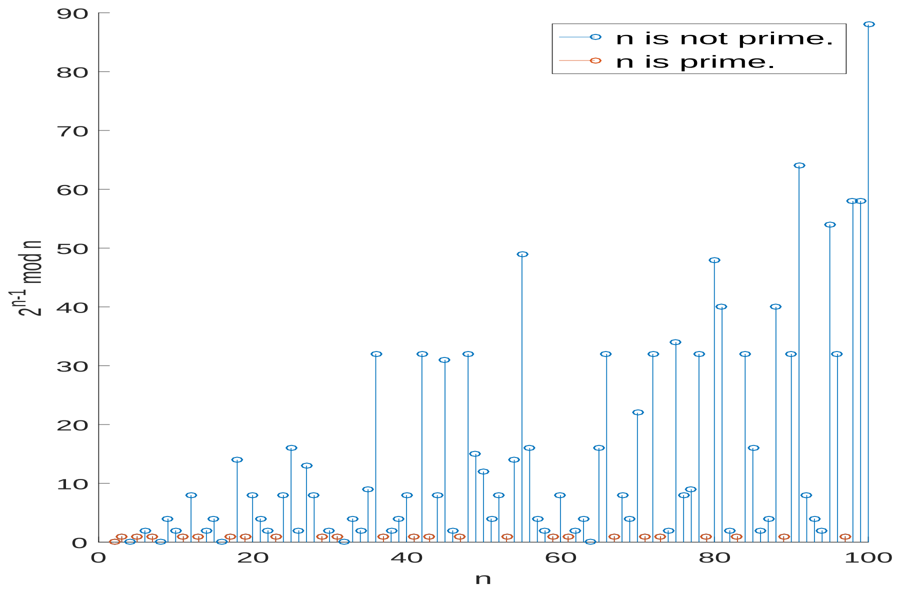
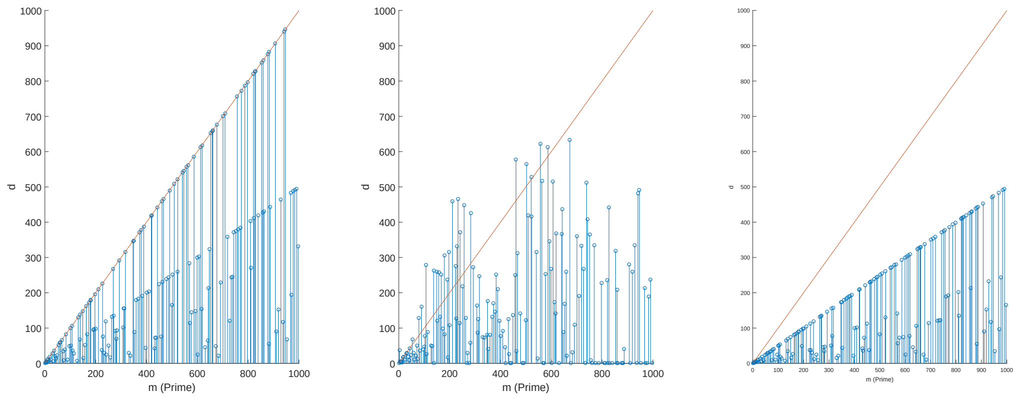


Figure 1: The distribution of the remainder of 2^{n-1} divided by n

Figure 1 is indeed a special case of Little Theorem of Fermat, which states that $m^{n-1} \equiv 1\,(mod\ n)$ when n is prime and m cannot be divided by n. Unfortunately, the inverse of Little Theorem of Fermat does not hold, i.e., the condition $m^{n-1} \equiv 1\,(mod\ n)$ does not necessarily mean that n is prime. Now we are going to figure out the relationship of d and m, where d is the least integer such that $n^d \equiv 1\,(mod\ n)$, where $n = 2, 3, 4$, $m = 1, 2, ..., 1000$, and m is prime. See Figure 2 for details.



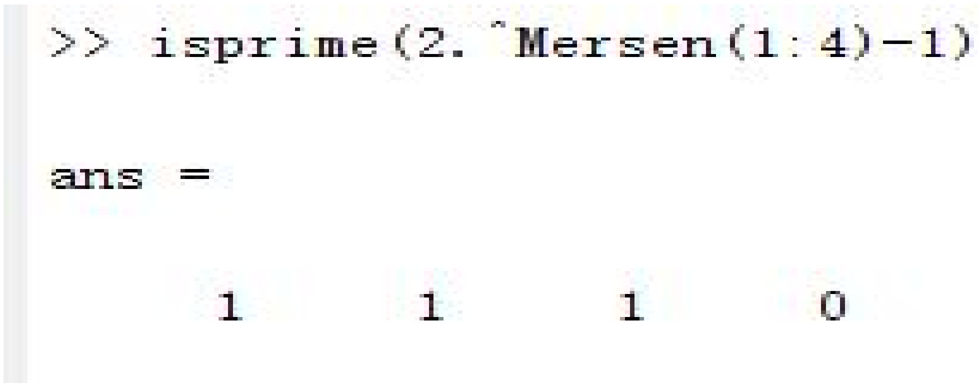
(a) n=2 (b) n=3 (c) n=4

Figure 2: The distribution of d and m

We know from the plots that the increase of d is $O(n)$. Hence it is time-consuming to check primality in this way.

Number theorists have proposed other efficient approaches to check primality of a number, like the $n-1$ test and probabilistic test. In fact, we have more efficient tests for the numbers with particular properties, like the well-known Mersenne numbers. With Matlab programmings, we know that the Mersenne prime number is sparse within all prime numbers. When n is less than 30, the Mersenne prime numbers are only $M_2 = 3, M_3 = 7, M_5 = 31, M_7 = 127, M_{13} = 8191, M_{17} = 131071, M_{19} = 524287$. Lucas-Lehmer primality test is a primality test exclusively for Mersenne numbers.[3] Define a sequence by $u_1 = 4, u_{k+1} = u_k^2 - 2\,mod\,M_n$, where $k = 1, 2, ..., n$. If $u_{n-1} \equiv 0\,mod\,M_n$, then M_n is a prime number. With the Lucas-Lehmer primality test, we know that M_7 is a prime number while $M_1 1$ not. See codes of Exercise 7 in the Appendix.

Based on the property of Mersenne prime numbers, one may hypothesize that M_n is prime when n is a Mersenne prime number. However, our experiment shows that this hypothesis does not hold when $n = M_4$. See Figure 3 for details.

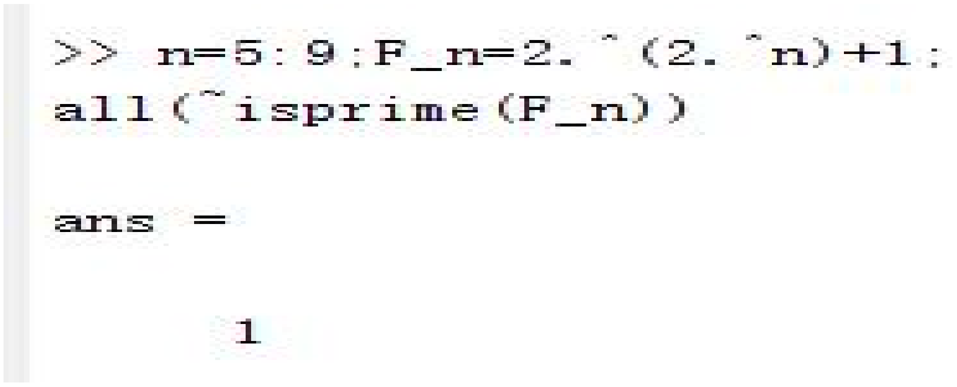


```
>> isprime(2.^Mersen(1:4)-1)

ans =

     1     1     1     0
```

Figure 3: The primality test of $2^{M_n} - 1$

2.2 Prime Generated Formula

In the preceding subsection, we have introduced some prime numbers generated algorithms and realized that it is not an easy task to obtain large prime numbers. Therefore, one may envision an omnipotent formula whose values are always prime numbers. Fermat, a French mathematician, pointed out that $F_n = 2^{2^n} + 1$ for all $n \in \mathbb{N}$. However, our Matlab programming shows, in effect, that F_n are all composite numbers when $n = 5, 6, 7, 8, 9$. See Figure 4 for details.



```
>> n=5:9;F_n=2.^(2.^n)+1;
all(~isprime(F_n))

ans =

     1
```

Figure 4: The primality test of F_n

Besides Fermat, Euler also proposed a univariate polynomial with integer coefficients, $n^2 + n + 41$. However, this formula only yields 7% of prime numbers that are less than 10000. Under the same settings, $n^2 - 79n + 1601$ accounts for 7.81% of such prime numbers while $6n^2 + 6n + 31$ only accounts for 2.93%.

Our experimental results give us a superficial understanding of the impossibility of such a univariate formula. Next we outline the proof of this proposition.

Proposition. *There exists no univariate polynomials $f(n)$ with integer coefficients such that for each $n \in \mathbb{N}$, $f(n)$ is prime.*

Proof. We prove this by contradiction. Assume that $f(n) = a_m n^m + a_{m-1} n^{m-1} + \cdots + a_1 n + a_0$, where $a_i \in \mathbb{Z}, i = 0, 1, ..., m$, is a formula that always yields prime numbers on integer points. Then there exists a p such that $f(p) \neq 0$. Consider $x = p + k \cdot f(p)$, where $k \in \mathbb{Z}$.

By the Binomial Theorem, we know that $x^j = [p + k \cdot f(p)]^j = p^j + f(p) \cdot M(j)$, where $M(j) \in \mathbb{Z}$. Hence $f(x) = a_m[p^m + f(p) \cdot M(m)] + \cdots + a_1[p + f(p) \cdot M(1)] + a_0 = f(p) + M \cdot f(p)$, which is the multiple of $f(p)$. □

As a consequence of the Hilbert's tenth problem, Matiyasevich proved that there exists an multivariate polynomial $P(x_1, x_2, ..., x_n)$ such that its positive values formulate the entire set of prime numbers.

2.3 Distribution of Prime Numbers

The distribution of prime numbers on the real line turns out to be abnormal. Let $\pi(n)$ denote the number of prime numbers less than n and $\pi(m, n)$ the number of prime numbers within $[m, n]$. Our experiments indicate that $\pi(100) = 25, \pi(1000) = 168, \pi(10000) = 1229, \pi(100000) = 9592$, while $\pi(100, 200) = 21, \pi(1000, 1100) = 16, \pi(10000, 10100) = 11, \pi(100000, 100100) = 6$. Thus, as n increases, the distribution of prime numbers become sparser.

Reorder prime numbers in the ascending order such that $p_1 = 2, p_2 = 3, ...$ Let $d_n = p_{n+1} - p + n$. We visualize the (p_n, d_n) on the plane. See Figure 5 for details.

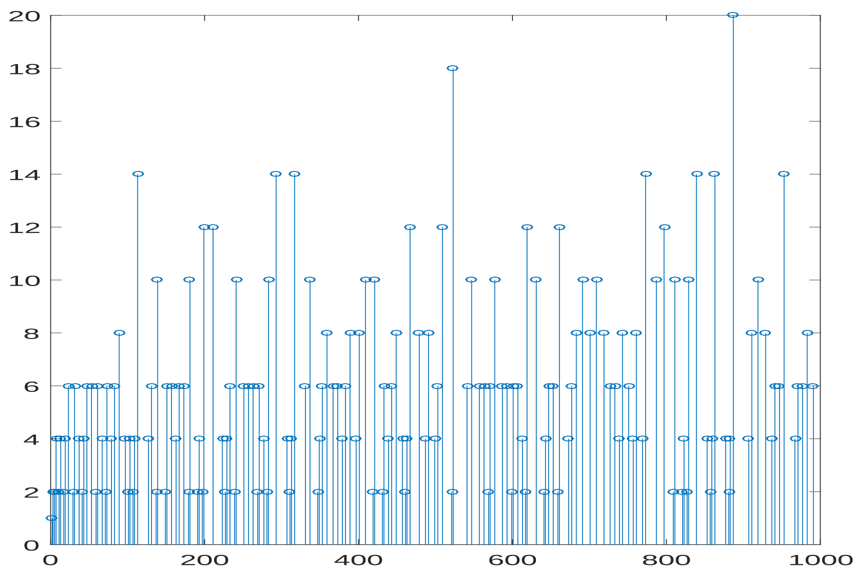


Figure 5: The Plotting of (p_n, d_n)

Not all the positive integers can work as the distance between two prime numbers. For instance, it should be even except 1. See Figure 6 for details.

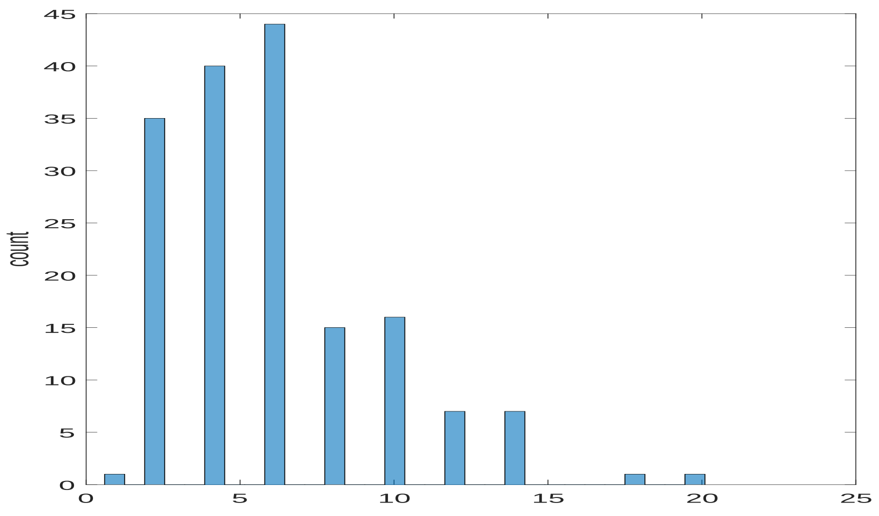


Figure 6: The Histogram of the distances between consecutive prime numbers

The distances of two consecutive prime numbers tend to take 6 or 8 as values. Moreover, the maximal distance between two consecutive prime numbers also increases as N increases. See Figure 7 for details.

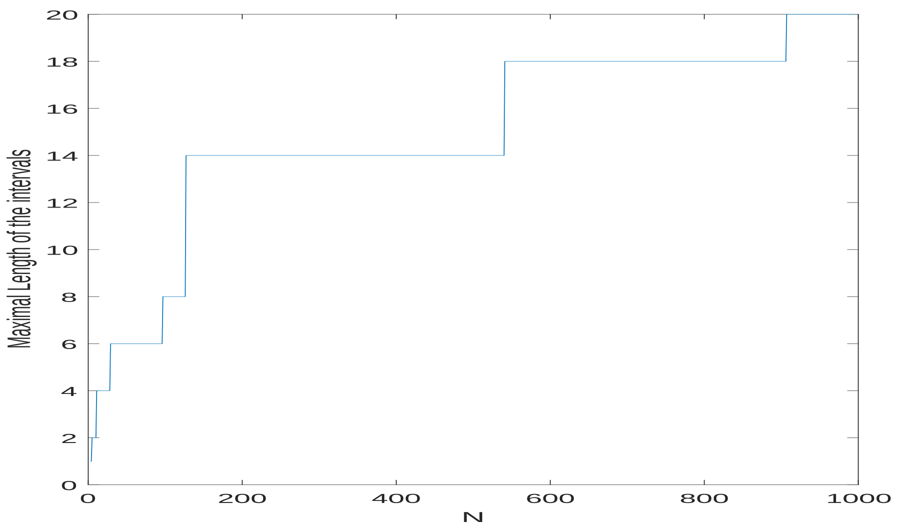


Figure 7: The maximal distances between two consecutive prime numbers

From this plot, one may claim that there are infinitely many consecutive prime numbers whose distances is any even integers. We delate the discussion of this hypothesis in the next section.

Also, we can plot $(n, \pi(n)), n = 1, 2, ..., N$ and inspect the growth rate of $\pi(n)$. See Figure 8 for details.

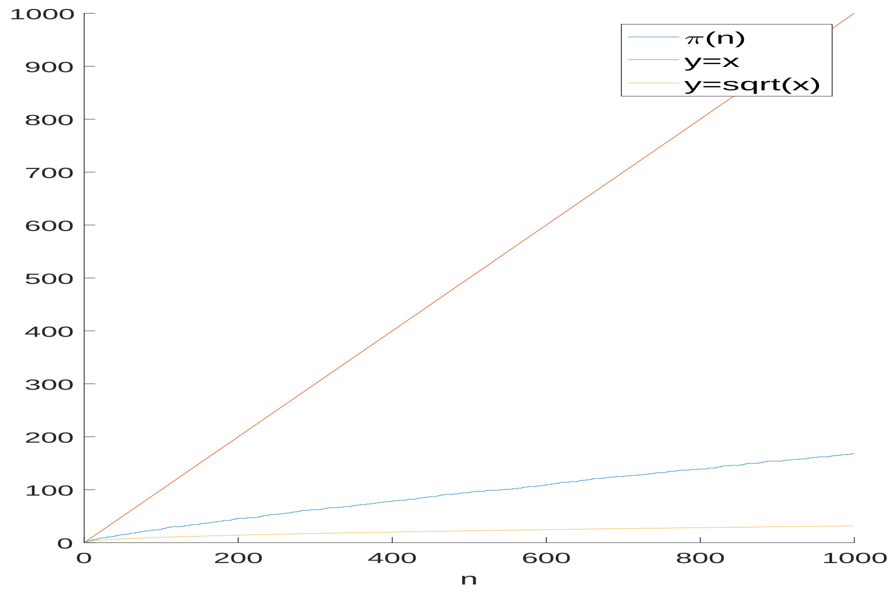


Figure 8: The growth rate of $\pi(n)$

When we make a plotting of $\pi(n)$ versus $\frac{n}{log\, n}$ the plotting turns out to be a straight line. Because of this, we may guess that the growth rate of $\pi(n)$ is $O(\frac{n}{log\, n})$. See Figure 9 for details.

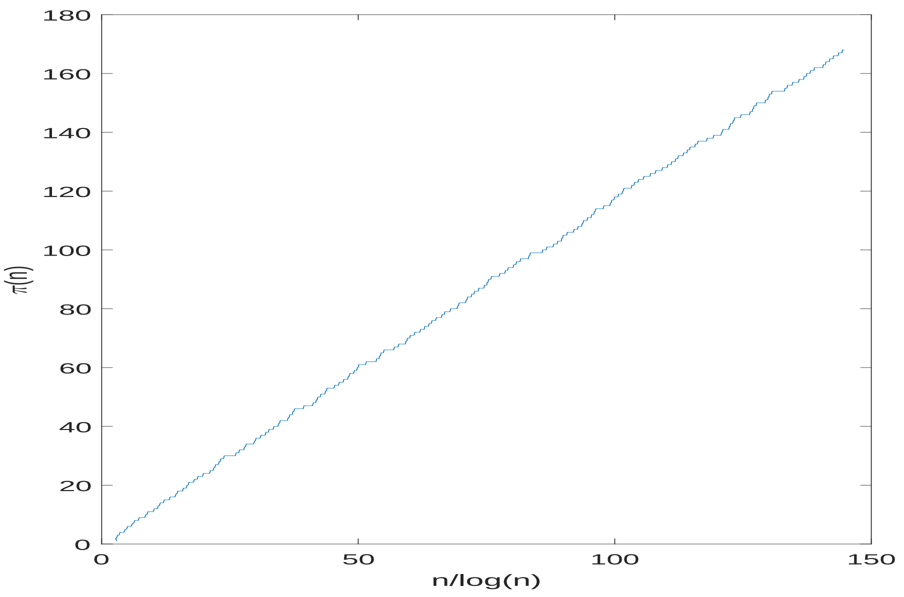


Figure 9: The plotting of $\pi(n)$ versus $\frac{n}{log\, n}$

Finally we describe some approximation formulas for $\pi(n)$. Let

$$Li(n) = \int_2^n \frac{1}{log\, x} dx,$$

$R(n) = 1 + \sum\limits_{k=1}^{\infty} \frac{(log\, n)^k}{k\zeta(k+1)\cdot k!}$, where $\zeta(k) = \sum\limits_{m=1}^{\infty} \frac{1}{m^k}$. See Figure 10 for details.

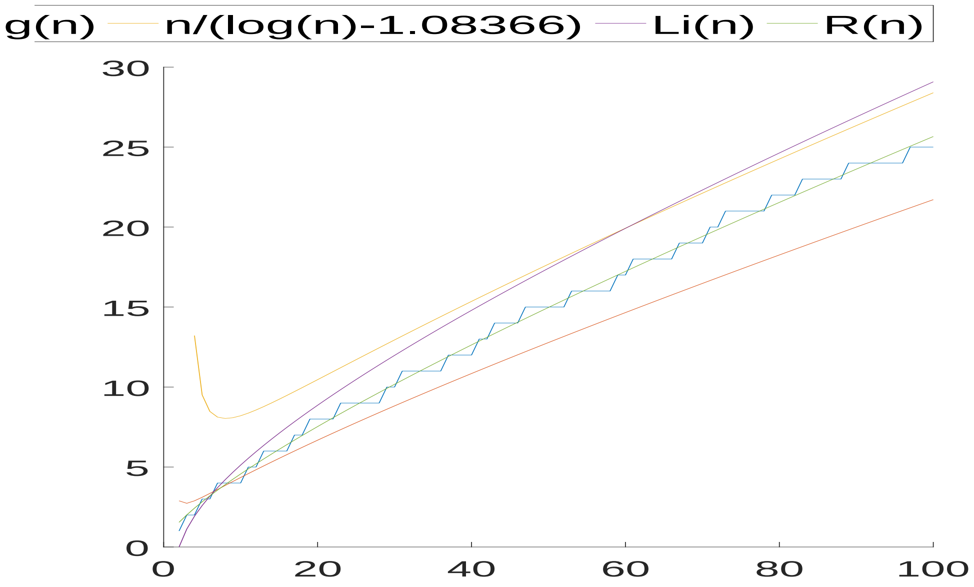


Figure 10: The comparison of $\pi(n)$, $\frac{n}{log(n)}$, $\frac{n}{log(n)-1.08366)}$, $Li(n)$, $R(n)$

From the comparative plot, we immediately notice that Riemman give the best approximation of $\pi(n)$ by formulating $R(n)$.

2.4 Further Problems

2.4.1 Goldbach Conjecture

In 1742, Goldbach claimed that Every even integer greater than 2 can be expressed as the sum of two primes. Our experiment shows that this conjecture is totally correct for all the even integers that are less than 10000. For instance, 19940=9973+9967. See codes in the Appendix.

2.4.2 Prime Factorization of Large integers

As the security foundation of the "RSA" cryptographic algorithm, the prime factorization of large integer is fundamentally difficult. For instance, no one can factor the Fermat number F_9. However, with inscribed function *factor()*, we can factor F_5 with no effort, which is 3, 5, 17, 257, 65537.

2.4.3 Perfect number

In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself.[4] With computer programming, we note that all perfect numbers within 10000 are 6, 28, 496, 8128. In addition, except for 6, all perfect numbers can be written as the cubic sum of some consecutive odd integers, such as $28 = 1^3 + 3^3, 496 = 1^3 + 3^3 + 5^3 + 7^3, 8128 = 1^3 + 3^3 + 5^3 + 7^3 + 9^3 + 11^3 + 13^3 + 15^3$.

2.4.4 Twin Prime Conjecture

Twin prime conjecture states that there are an infinite number of pairs of twin primes. In 2013, Chinese mathematician Yitang Zhang managed to prove that for some integer N that is less than 70 million, there are infinitely many pairs of primes that differ by?N. The bounded gaps between primes was improved to be 246 by Professor Terence Tao from UCLA.

3 Conclusion and Discussion

In this experiment we implement the sieve of Eratosthenes and tentative division methods to obtain all the prime numbers up to a given limit. Later, by analyzing Mersenne numbers, we have a preliminary understanding of the difficulty of generating large primes. After the failure of formulating a univariate polynomial with integer coefficients, which can generate prime numbers, we turn to investigating the distribution of prime numbers. All these experiments reveal the mystery within the realm of number theory. Those open questions require mathematician to dive more deeply into them generation after generation.

4 Reference

[1] Wikipedia ***Prime number.*** URL: `https://en.wikipedia.org/wiki/Prime_number` Retrieved 19 December, 2017.

[2] Shangzhi Li, Falai Chen, Yaohua Wu, and Yunhua Zhang (1999) ***Mathematical Experiments.*** Textbook Seris for 21st Century, Higher Education Press.

[3] Wikipedia ***LucasCLehmer primality test.*** URL: `https://en.wikipedia.org/wiki/Lucas-Lehmer_primality_test` Retrieved 19 December, 2017.

[4] Wikipedia ***Perfect number.*** URL: `https://en.wikipedia.org/wiki/Perfect_number` Retrieved 19 December, 2017.

5 Appendix

```
% Exercise 1
N=100;primeNum=2:N;
for i=1:N
    temp=primeNum(i);
    Notprime=(mod(primeNum,temp)==0);
    if sum(Notprime)==1
        break;
    else
        Notprime=find(Notprime);
        Notprime=Notprime(2:end);
        index=zeros(1,length(primeNum));
        index(Notprime)=1;
        primeNum=primeNum(~logical(index));
    end
end
primeNum

% Exercise 2
function primeNum = PRIME(N)

primeNum=2:N;
for i=1:N
    a=primeNum(i);
    Notprime=(mod(primeNum,a)==0);
    if sum(Notprime)==1
```

```
            break;
        else
            Notprime=find(Notprime);
            Notprime=Notprime(2:end);
            index=zeros(1,length(primeNum));
            index(Notprime)=1;
            primeNum=primeNum(~logical(index));
        end
end

end

N=10000;
seivePrime=PRIME(N);

% Exercise 3
function Prime2 = DIVprime(N)

Prime2=zeros(1,N);
Prime2(1)=2; i=2;
for m=3:2:N
    cri=mod(m, Prime2(Prime2~=0));
    if all(cri)
        Prime2(i)=m;
        i=i+1;
    end
end
Prime2=Prime2(Prime2~=0);

end

N=10000;
tic
divPrime=DIVprime(N);
tD=toc;
tic
seivePrime=PRIME(N);
tS=toc;

% Exercise 4
N=100;mod2n_1=zeros(1,N-1);m=2;
for n=2:N
    mod2n_1(n-1)=mod(m^(n-1),n);
end
hold on
stem(2:N, mod2n_1), xlabel('n'), ylabel('2^{n-1}␣mod␣n')
cri=PRIME(N)-1;
all(mod2n_1(cri(m:end))==1)
stem(cri+1,mod2n_1(cri))
legend('n␣is␣not␣prime.','n␣is␣prime.')
hold off
print('Ex4','-dpdf','-fillpage')

% Exercise 5
n=4;N=1000;d=1;
m=1:N; remainder=zeros(1,N);
times=1;
while any(~remainder(PRIME(N))) & times<10000
    remainder(mod(n^d,m)==1 & ~remainder)=d;
    d=d+1;
    times=times+1;
end
hold on
stem(PRIME(N),remainder(PRIME(N))),xlabel('m␣(Prime)'),ylabel('d')
plot(1:N,1:N)
hold off
sum(remainder == (1:N)-1)
print('Ex51','-dpdf','-fillpage')
print('Ex52','-dpdf','-fillpage')
print('Ex53','-dpdf','-fillpage')

% Exercise 6
N=30;Mersen=2.^(2:N)-1;
Mersen=Mersen(isprime(Mersen))
all(isprime(2.^PRIME(100)-1))

% Exercise 7
n=7;M_n=2^n-1;u=zeros(1,n);
u(1)=4;
for i=2:n
    u(i)=mod(u(i-1)^2-2,M_n);
end
u(n-1)==0

n=11;M_n=2^n-1;u=zeros(1,n);
u(1)=4;
for i=2:n
    u(i)=mod(u(i-1)^2-2,M_n);
end
u(n-1)==0

% Exercise 8
isprime(2.^Mersen(1:4)-1)

% Exercise 9
n=5:9;F_n=2.^(2.^n)+1;
all(~isprime(F_n))

% Exercise 10
formula=@(n) n.^2+n+41;
n=0:100;
all(isprime(formula(n)))
intereP=formula(n);
sum(isprime(intereP(intereP <10000)))/length(PRIME(10000))

formula=@(n) n.^2-79*n+1601;
n=0:100;
all(isprime(formula(n)))
intereP=formula(n);
sum(isprime(intereP(intereP <10000)))/length(PRIME(10000))

formula=@(n) 6*n.^2+6*n+31;
n=0:100;
all(isprime(formula(n)))
intereP=formula(n);
sum(isprime(intereP(intereP <10000)))/length(PRIME(10000))

% Exercise 12
```

```
n=[100 1000 10000 100000];
Pi_n=zeros(1,4);Pi_n1_n2=zeros(1,4);
for i=1:4
    Pi_n(i)=length(PRIME(n(i)));
    Pi_n1_n2(i)=length(PRIME(n(i)+100))-length(PRIME(n(i)));
end
Pi_n
Pi_n1_n2


% Exercise 13
N=1000;
prim=PRIME(N);
d_n=prim(2:end)-prim(1:(end-1));
stem(prim(1:(end-1)),d_n)
print('Ex131','-dpdf','-fillpage')
unique(d_n)
histogram(d_n,30),ylabel('count')
print('Ex132','-dpdf','-fillpage')
max_inter=zeros(1,N-2);
for n=3:N
    prim=PRIME(n);
    d_n=prim(2:end)-prim(1:(end-1));
    max_inter(n-2)=max(d_n);
end
plot(3:N,max_inter),xlabel('N'),ylabel('Maximal Length of the intervals')
print('Ex133','-dpdf','-fillpage')

% Exercise 15
N=1000;Pi_n=zeros(1,N);
for n=2:N
    Pi_n(n)=length(PRIME(n));
end
hold on
plot(1:N,Pi_n),xlabel('n')
plot(1:N,1:N)
plot(1:N,sqrt(1:N))
legend('\pi(n)','y=x','y=sqrt(x)')
hold off
print('Ex151','-dpdf','-fillpage')
plot(1:N,Pi_n./sqrt(1:N))
plot((1:N)./log(1:N),Pi_n),xlabel('n/log(n)'),ylabel('\pi(n)')
print('Ex152','-dpdf','-fillpage')

% Exercise 16
syms k x
N=100;Pi_n=zeros(1,N-1);
Li_n=zeros(1,N-1);R_n=zeros(1,N-1);
for n=2:N
    Pi_n(n-1)=length(PRIME(n));
    f=@(x) 1./log(x);
    Li_n(n-1)=integral(f,2,n);
    R_n(n-1)=1+symsum(log(n)^k/(k*zeta(k+1)*factorial(k)),k,1,100);
end
hold on
plot(2:N,Pi_n)
plot(2:N,(2:N)./log(2:N))
plot(4:N,(4:N)./(log(4:N)-1.08366))
plot(2:N,Li_n(2:end))
plot(2:N,R_n(2:end))
legend('\pi(n)','n/log(n)','n/(log(n)-1.08366)','Li(n)','R(n)','Location','northoutsi
hold off
norm(Pi_n(end)-N/log(N))
norm(Pi_n(end)-N/(log(N)-1.08366))
norm(Pi_n(end)-Li_n(end))
norm(Pi_n(end)-R_n(end))
print('Ex16','-dpdf','-fillpage')

% Extra 1 Goldbach Conjecture
N=10000;Gold_num=zeros(N/2-2,3);
prim=PRIME(N);
prim=prim(2:end);
for n=3:N/2
    Gold_num(n-2,1)=2*n;
    for i=1:length(prim)
        for j=1:length(prim)
            if 2*n==prim(i)+prim(j)
                Gold_num(n-2,2)=prim(i);
                Gold_num(n-2,3)=prim(j);
            end
        end
    end
end
Gold_num

% Extra 2 Integer Factorization
factor(2^(2^5)-1)

% Extra 3 Perfect number
i=1;N=10000;Perf_num=zeros(1,N);
for n=2:N
    if sum(divisors(n))-n==n
        Perf_num(i)=n;
        i=i+1;
    end
end
Perf_num=Perf_num(logical(Perf_num));
Perf_num
% 6          28          496          8128
496==1^3+3^3+5^3+7^3
8128==1^3+3^3+5^3+7^3+9^3+11^3+13^3+15^3
```