5J-05

# ルジャンドル予想の数値的検証

山口　博将 †　　高橋　大介 ‡
筑波大学大学院理工情報生命学術院 †　筑波大学計算科学研究センター ‡

## 1　研究の目的

昨今に至ってもなお未解決な問題は無数に存在する．その中でも素数に関連する問題は，素数が無数に存在することから，解明するのが難しい問題ばかりである．

一方，計算機というのは，様々な分野の問題を解明し続けているとともに，多くの人の手によって日々改良を重ね急速に発展しており，未解決問題についても計算機を用いて正しさを確かめる試みが行われてきた．

そこで，本研究では素数に関する未解決問題の一つである，ルジャンドル予想を取り上げる．ルジャンドル予想とは，「任意の自然数 $n$ について $n^2$ と $(n+1)^2$ の間に必ず素数が存在する」という予想である．Legendre によって提唱された，現在でも解明されていない問題の一つである．本研究では，これまでに網羅的に調べられた $4 \times 10^{18}$ より大きな自然数に対して，計算機を用いて素数判定を行い，ルジャンドル予想の反例を探すことを目的とする．

関連研究として，1975 年に Chen[1] は，任意の自然数 $n$ について，区間 $[n^2, (n+1)^2]$ に必ず素数または半素数があることを示した．1937 年に Ingham[2] は，$n$ が十分に大きい場合 $[n^3, (n+1)^3]$ に必ず素数があることを示した．ゴールドバッハ予想は，「全ての 2 よりも大きな偶数は 2 つの素数の和として表すことができる」という予想で，ルジャンドル予想と同じく素数を扱う未解決問題である．この予想に関するプロジェクトで，素数については $4 \times 10^{18}$ まで網羅的に調べられている [3].

## 2　実装方針

引数として与える自然数 $j$ までのすべての自然数について，ルジャンドル予想を確かめるために **Algorithm 1** のように考えた．これは $k = n^2 + 1$ から $k$ をインクリメントしつつそれぞれ素数判定を行う．primetest($k$) では $k$ についての素数判定を行う．$k$ が素数であれば true を返し，次の $n$ のループの判定に入る．もしその $n$ について $k = (n+1)^2 - 1$ まで素数判定を行っても素数が見つからなかった場合，反例の検出を出力する．本研究では，Miller–Rabin 素数判定法 [4, 5] を用いて primetest($k$) を実装した．

### 2.1　Miller–Rabin 素数判定法

Miller-Rabin 素数判定法とは，Miller[4] によって開発された Miller テストを Rabin[5] が確率的なアルゴリズムとして改良した確率的素数判定法の一つで，本研究では多くの自然数を高速に判定するために用いる．

Numerical Verification of the Legendre's conjecture using Miller–Rabin primality test.

† Hiromasa Yamaguchi, Graduate School of Science and Technology, University of Tsukuba
‡ Daisuke Takahashi, Center for Computational Sciences, University of Tsukuba

**Algorithm 1** Confirmation of Legendre's conjecture for numbers up to $j$

```
function LEGENDRE(j)
    for n = 1 to j do
        k ← n^2 + 1
        end ← (n + 1)^2
        while k < end do
            if primetest(k) then
                break
            k ← k + 1
        if k = end then
            print("There is no prime.")
    return
```

**Algorithm 2** は，Miller–Rabin 素数判定法のアルゴリズムである．$B[i]$ は Miller–Rabin 素数判定法の計算で用いられるパラメータであり，$2^{64}$ までの素数については決定論的かつ効率よく計算できるパラメータが存在する [6]．剰余乗算 $B[i]^s \mod n$ を求める関数 PowMod($B[i], s, n$) では剰余を求める除算命令がボトルネックとなっている．

**Algorithm 2** Miller–Rabin primality test

```
Require: n ∈ ℕ, B:Array of Base parameter,
         k : Length of B
Ensure: Result : n is composite,
        or probably prime
  function MRTEST(n,k)
    d ← n − 1
    while d mod 2 = 0 do
        d ← d/2
    for i = 0 to k do
        s ← d
        y ← PowMod(B[i], s, n)
        for j = 1 to s do
            y ← (y ∗ y) mod n
            if y = n − 1 then
                return false;
            if y = 1 then
                return true;
    return true;
```

### 2.2　Montgomery 乗算アルゴリズム

Montgomery 乗算アルゴリズム [7] は，$N$ と $R$ が互いに素で $N < R$ を満たす自然数であるとき，$NN' \equiv -1 \mod R$ を満たす $N'$ を用いて整数 $A(< N)$ に対し，$\mathrm{MR}(A) = AR^{-1} \mod N$ を求める Montgomery リダクションと呼ばれる演算を用いることで乗算の計算を行うアルゴリズムである．あらかじめ $R^2 \mod N$ を計算しておくことで，Montgomery リダクションを用いて整数 $a$ の Montgomery 表現 $A$ は $A = \mathrm{MR}(aR^2) = aR \mod N$ として得られる．Montgomery 表現の状態で乗算を行うと，$A \times A \equiv a^2R^2 \mod N$ となり $\mathrm{MR}(A \times A) = a^2R \mod N$ として Montgomery リダクションを行うことで $a^2$ の Montgomery 表現が得られる．本研究では，このように Montgomery リダクションを複数

回用いることによって剰余乗算 $a^e \mod N$ を頻繁に計算している．

また，Montgomery 乗算アルゴリズムに用いる $R$ を 2 のべき乗にすることで，剰余演算は $R-1$ とのビットマスクで計算でき，除算命令はシフト演算に置き換えることができる．本研究では，Miller–Rabin 素数判定法で判定する整数 $N$ は必ず奇数となっているため，$N$ が奇数である場合に $N$ と $R$ は互いに素であるという条件を満たしながら，$R$ が 2 のべき乗である場合の Montgomery 乗算アルゴリズムを用いて剰余乗算 $a^e \mod N$ を計算することができる．

あらかじめ $NN' \equiv -1 \mod R$ を満たす $N'$ と $R^2 \mod N$ を計算しておくことで，調べる整数 $N$ 一つにつき除算命令は 1 回で $N$ の素数判定が計算可能になった．

**Algorithm 3** Montgomery Reduction

```
function MR(T)
    m ← (T mod R)N' mod R
    t ← (T + mN)/R
    if t ≥ N then
        t ← t − N
    return t;
```

## 3 実装と結果

### 3.1 OpenMP による並列化

本研究では，OpenMP でプログラムを並列化し，68 コア 272 スレッドのメニーコアプロセッサである Intel Xeon Phi 7250 を用いて計算した．図 1 は $n$ が $10^5$, $10^6$, $10^7$ までのルジャンドル予想の検証に要した時間とスレッド数の相関をプロットしたものである．$n = 10^5$ については計算量が少ないため，計算時間全体に対して並列化によるオーバーヘッドが占める時間が相対的に大きくなっていることから，スレッド数を増やしても速度向上率が大きく変わらなかったと考えられる．一方，$n = 10^6$, $10^7$ の場合から，$n$ が大きくなるに従って，並列化したことによる大幅な速度向上が確認できる．したがって，OpenMP を用いて並列化し，272 スレッドのメニーコアプロセッサで計算することにより，さらに大きな $n = 2^{32}$ の場合はより大きな速度向上が得られていると考えられる．

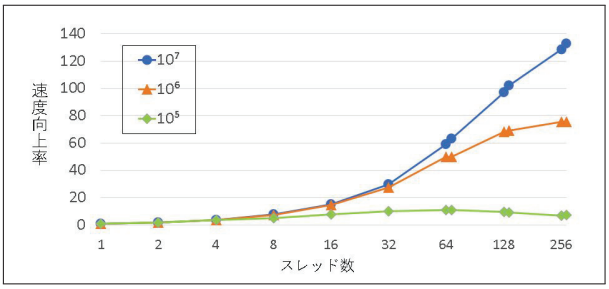


図 1 スレッド数と実行速度の相関

### 3.2 Montgomery 乗算アルゴリズムによる高速化

本研究では，ルジャンドル予想の $n$ について $2^{32}-1$ まで判定した．また，Montgomery 乗算アルゴリズムを実装し，実装前と実装後の実行時間を測定した．表 1 は，Montgomery 乗算アルゴリズムを用いる前後の実行結果である．

表 1 ルジャンドル予想の検証に要した実行時間 (sec)

| $n$ | $10^8$ | $2 \cdot 10^9$ | $2^{32}-1$ |
|---|---|---|---|
| Montgomery 乗算なし | 37.869 | 1099.464 | 2534.137 |
| Montgomery 乗算あり | 10.334 | 299.468 | $N/A$ |
| 速度向上率 | 3.665 | 3.671 | – |

Montgomery 乗算アルゴリズムを用いた場合，$n = 2^{32}-1$ は測定できていない．これは Montgomery 乗算アルゴリズムを実装するにあたって，$n = 2^{32}-1$ を引数に与えると計算過程で符号なし 128bit 整数型ではオーバーフローしてしまう場合があるためである．

$n = 2^{32}-1$ の場合において反例の出力がなかったことから，ルジャンドル予想は自然数 $n$ について区間 $[1, 2^{32}]$ において正しさが確かめられたといえる．

$n = 10^8$, $n = 2 \cdot 10^9$ のどちらの場合においても，約 3.67 倍ほどの性能向上が見られた．これは Montgomery 乗算アルゴリズムを適用することで除算命令を減らすことができているためであり，素数判定 1 回にかかる時間が短縮されていると考えられる．したがって，$n = 2^{32}-1$ についても計算可能な Montgomery 乗算アルゴリズムを用いたプログラムを実装し実行した場合，同様に速くなると考えられる．

## 4 まとめ

$2^{64}$ までの整数について Miller–Rabin 素数判定法を用いてそれぞれ判定し，OpenMP を用いてメニーコアプロセッサで並列計算することによって，区間 $[1, 2^{32}]$ の $n$ についてルジャンドル予想を数値的に検証し，反例がないことを確かめた．また，$R$ を 2 のべき乗とした Montgomery 乗算アルゴリズムを実装することで除算命令を実質的に減らし，ルジャンドル予想の $n = 2 \cdot 10^9$ までの検証において速度向上を達成した．

## 参考文献

[1] Chen Jing-Run. On the distribution of almost primes in an interval. *Sci. Sin.*, Vol. 18, pp. 611–627, 1975.

[2] Ingham A., E. On the difference between consecutive primes. *Q. J. Math. Oxford*, Vol. 8, No. 1, pp. 255–266, 1937.

[3] Andersen J., K. Maximal prime gaps. URL `http://primerecords.dk/primegaps/maximal.htm`.

[4] Miller G., L. Riemann's hypothesis and tests for primality. *J. Comput. Syst. Sci.*, Vol. 13, pp. 300–317, 1976.

[5] Rabin M., O. Probabilistic algorithm for testing primality. *J. Number Theory*, Vol. 12, pp. 128–138, 1980.

[6] Deterministic variants of the miller-rabin primality test. URL `https://miller-rabin.appspot.com/`.

[7] Montgomery P. Modular multiplication without trial division. *Math. Comput.*, Vol. 44, pp. 519–521, 1985.