ハイパフォーマンス
コンピューティング 86－1
（２００１．５．２５）

# 高精度数の高速基数変換法とその性能

平山 弘　菅家 英和

神奈川工科大学

高精度数の基数変換、特に 2 進 10 進数の変換は実用上非常に重要である。この変換については、平山および後等によって高速計算法が提案されている。本論文では、実際に高精度数の 2 進 10 進数の変換を行いその性能を評価した。Pentium Ⅲ 933MHz を使用し、Borland C++言語を使って、いろいろな桁数の高精度数の基数変換を行った。この結果、高精度数が約 500 桁以上の数値であるならば、従来から知られている計算法より高速に計算できることがわかった。1 万桁の数値の変換では 3.6 倍、10 万桁の数値では、17.9 倍高速になる。2 進 10 進数変換の演算は、同じ桁数の平方根の計算より 2 倍程度時間がかかるもわかった。

# Fast Radix Conversion Method of High Precision Numbers and its Performance

Hiroshi Hirayama and Hidekazu Kanke

Kanagawa Institute of Technology

The radix conversion of the high precision number, especially, the conversion of the binary number to decimal number is very important in case of practical use.As for this conversion, high-speed method is proposed by Hirayama and Ushiro et al.

In this paper, the efficiency of this method is tested by the numerical experiment which converts a high precision binary number into the decimal number.Using Pentium III 933 MHz and the Borland C++ Builder, a high precision binary number is converted nto the decimal number.

As a result, it found that it was possible to convert at higher speed than the usual method if applying this high-speed conversion method to the equal to or more than about 500 digit numerical value. By using the fast radix conversion method, the 10,000 digit and and the 100,000 digit numerical value can be converted 3.6 times and 17.9 times faster than usual one rspectively. As for the calculation which converts a binary number into the decimal number, it found that it took about twice time than the computation of the square root at the same number of the figures.

## 1．はじめに

級数の高精度計算を有理数を使って計算することによって、計算速度を上げる方法が多くの人によって使われている。この方法を使って、1989 年に Chudnovsky 兄弟による円周率 10 億桁の計算が有名である。しかしながら、彼らは、この計算方法の詳細を発表していない。

この方法は最近（1998 年、1999 年）にかけて、右田等[1]や後等[2]によって、級数計算への適用の再現が行われている。

この計算法の起源を調べると 1997 年にすでに B.Haible and T.Papanikolaou[4] によって公表されていることがわかる。この論文では、トーナメント法とか縮約法とか呼ばれているこれらの計算方法を Binary Splitting Algorithm（以降 BSA 法と略す）と呼んでいる。

この論文を読むと BSA 法は、1976 年に R. P. Brent、1987 年に Borwein 兄弟によって公表され、Chudnovsky 兄弟による円周率 10 億桁の計算が行われた 1989 年当時でもよく知られた方法であったことがわかる。

この方法は、平山[3]や後等[2]で述べられているように、２進数を１０進数などに変換する基数変換の高速計算法も与える。この計算法は計算する桁数を $P$ として、十分大きな数である場合、$O(p(\log p)^2)$ の計算量となる計算法である。この基数変換は、数式処理の結果などを高精度の数値を出力する場合、非常に重要である。

本論文では、理論で示された基数変換を、実際に行い、その性能を調べた。

以下の計算には、コンパイラとして、Borland C++ Builder 5.0、計算機として自作のパーソナル・コンピュータ(Pentium Ⅲ 933MHz)を利用した。

## 2．計算方法

実数 $a$ は、$r$ 進数で

$$a=\sum_{k=0}^{\infty} a_k r^{N-k} \qquad 0 \le a_k < r \tag{2.1}$$

と表現されます。これは、次の形式に変形できる。

$$\begin{aligned} a &= a_0 r^N + a_1 r^{N-1} + a_2 r^{N-2} + \cdots \\ &= r^N (a_0 + \frac{1}{r}(a_1 + \frac{1}{r}(a_2 + \frac{1}{r}(a_3 + \cdots \end{aligned} \tag{2.2}$$

この級数部分を行列の乗算形式で書くと

$$\begin{pmatrix} 1 & P_n \\ 0 & R_n \end{pmatrix} = \begin{pmatrix} 1 & a_0 \\ 0 & r \end{pmatrix} \begin{pmatrix} 1 & a_1 \\ 0 & r \end{pmatrix} \begin{pmatrix} 1 & a_2 \\ 0 & r \end{pmatrix} \cdots \begin{pmatrix} 1 & a_n \\ 0 & r \end{pmatrix} \tag{2.3}$$

となる。この行列の式(3)を計算し、

$$a = r^N \frac{P_n}{R_n} \tag{2.4}$$

として、$a$ が求めらる。この計算を 10 進数表現の多倍長演算プログラムで計算すると、10 進数に変換できることになる。(2.3)の式は、

$$\left\{ \begin{pmatrix} 1 & a_0 \\ 0 & r \end{pmatrix} \begin{pmatrix} 1 & a_1 \\ 0 & r \end{pmatrix} \right\} \left\{ \begin{pmatrix} 1 & a_2 \\ 0 & r \end{pmatrix} \begin{pmatrix} 1 & a_3 \\ 0 & r \end{pmatrix} \right\} \cdots \left\{ \begin{pmatrix} 1 & a_{n-1} \\ 0 & r \end{pmatrix} \begin{pmatrix} 1 & a_n \\ 0 & r \end{pmatrix} \right\} \tag{2.5}$$

と２つの行列を組み合わせかけ算を行う。さらにそれらを同じようにかけ算することによってかけ算を行う。トーナメント方式に隣同士の行列のかけ算を行う。このような操作を行うと途中の計算桁数を押さえることができる。また、計算桁数が大きくなったとき、高速 Fourier 変換を利用した高速乗算法が使えるため、この計算は高速に計算できる。このときの計算量[1]は、計算精度を $p$ とすると(2.1)の $a_k$ がある程度以上増大しないため収束が速くなり $O(p(\log p)^2)$ となる。

逆に、基数を $r=10$ にして、2 進数表現を使う多倍長計算プログラムで計算すると、１０進数を 2 進数に変換することになる。このときも、2 進数 10 進数変換と同様に、$O(p(\log p)^2)$ の計算量で計算できる。すなわち、$O(p(\log p)^2)$ の計算量で基数を変換できることになる。この計算法は一般に、他の基数に変換する場合にも同じことが言える。

## 3. 多倍長計算プログラム

多倍長計算プログラムとして、既存の 10000 進数を使った多倍長計算プログラム[5]に FFT を使った乗算機能を付加し、計算精度を可変に出来るように変更した。FFT のプログラムとして実数型 FFT 計算法として有名な Bergland[6]のアルゴリズムを使用した。

さらに、この多倍長計算プログラムを 10000 進数から $2^{32}$ 進数に変更した。$2^{32}$ 進数への変更は、高級言語だけを使用しても可能であるが、高級言語だけ使用した場合、メモリーの使用効率は若干良くなるが、速度がかなり低下するので、プログラムの一部にインライン・アセンブラを使用した。このため、$2^{32}$ 進数を使う多倍長計算プログラムは、計算機およびコンパイラーに依存することになる。使用できる機種は、インテル社の 32 ビットの計算機だけとなる。インライン・アセンブラ記述方法は、Borland 社の C++ Builder 用にしたが、この記述方法は、Microsoft 社の Visual C/C++と非常に似ており、このプログラムでは、この二つのコンパイラーで共通化することができた。

インライン・アセンブラを使用すると、加減算にキャリー付きの加減算命令を使用でき、符号なしの３２ビットの同士の整数をかけ算を行い６４ビットの計算結果を出すことができるので、効率的に計算することができる。１０進数に換算して 1000 桁程度の計算では、約５倍程度の計算速度の向上が見られた。

$2^{32}$ 進数を使う多倍長計算プログラムでは、32 ビットの数値を２個の 16 ビットの値に分けて、FFT プログラムに送り、その計算結果を利用して乗算や２乗計算を行っている。この時、倍精度に入る数値が最大で 65535 となる。このため、倍精度（６４ビット）で作った FFT プログラムでは、大きな精度の計算ができなくなるので、拡張精度（８０ビット）を利用したをプログラムを使用した。拡張精度も計算機に依存する計算精度であるが、他の部分も計算機に依存するので新しく問題が生じることはない。

非常に桁数の大きい数値の乗算に使われる FFT を利用した乗算法では、$2^{32}$ 進数を使うメリットは少ない。メモリーの使用効率は少し向上するが、計算時間の大半を占める FFT プログラムは高級言語で作られており、計算速度向上する部分はほとんどないので、1000 桁程度の計算のような速度向上は期待できないためである。

## 4.数値実験

数値実験として、$\sqrt{3}$の数値の２進数を１０進数に変換する計算を行った。$\sqrt{3}$は不規則な数値の例として使った。この結果を以下に示す。平方根の計算は、$\sqrt{3}$の平方根すなわち$\sqrt[4]{3}$を計算する時間である。この結果を見ると、10 進数で約 500 桁以上を越える数値では、本方法（BSA 法）が効果的であることがわかる。500 桁の数値は FFT の乗算法が効果的である桁数（約 1000～2000 桁）を下回る桁数である。この効果は計算途中の計算桁数を小さくすることによってなされた高速化であると考えられる。10 万桁や 20 万桁では、17.9 倍、26.2 倍とかなり高速になるが、これは FFT を利用した効果だと考えられる。

表１　２進数を１０進数に変換に要する時間（秒）

| 桁数（10 進数） | BSA 法による時間（秒） | 従来の方法による時間（秒） | 従来の方法/BSA 法による計算法 | 平方根の計算時間（秒） |
|---|---|---|---|---|
| 200000 | 2.954 | 77.531 | 26.2461 | 1.343 |
| 100000 | 1.094 | 19.578 | 17.8956 | 0.500 |
| 50000 | 0.375 | 4.9845 | 13.2920 | 0.172 |
| 20000 | 0.146 | 0.8593 | 5.8858 | 0.0677 |
| 10000 | 0.0656 | 0.2344 | 3.5732 | 0.0312 |
| 5000 | 0.0297 | 0.0672 | 2.2626 | 0.0141 |
| 2000 | 0.00814 | 0.0141 | 1.7272 | 0.00532 |
| 1000 | 0.00281 | 0.00407 | 1.4466 | 0.00219 |
| 500 | 0.000968 | 0.00116 | 1.1942 | 0.000688 |
| 200 | 0.000298 | 0.000281 | 0.9723 | 0.000172 |
| 100 | 0.000156 | 0.000138 | 0.8809 | 0.000094 |
| 50 | 0.000091 | 0.000077 | 0.8537 | 0.000060 |
| 20 | 0.000064 | 0.000055 | 0.8584 | 0.000042 |

平方根の計算も計算桁数が大きければ$O(p(\log p)^2)$の計算量で計算できる。ここでは他の計算との比較のために、平方根の計算時間を示す。表でもわかるように、平方根の計算が２進数と 10 進数に変換する時間より高速に行うことができる。２進数を１０進数に変換する演算は、平方根の計算より初等的で簡単に思えるが、計算時間は２進 10 進変換の計算の方が多くかかることがわかる。

この結果は、汎用の多倍長計算プログラムで行った実行時間である。細かい改良をすれば、かなり高速化する可能性がある。実際、この計算結果を出す場合でも、小さな改良かなり計算時間を左右する場合があった。たとえば、符号なし整数を 10000 進数の数値に変換する場合、常に正の数値しか扱わないので、符号処理を省くとかなり高速になった。

## 5. おわりに

２進数を１０進数変換などの基数変換の高速アルゴリズムの効果を調べた。約500桁以上の桁数の数値に対して有効で、桁数が大きくなるほど効果的である。この計算は平方根の計算と同じ$O(p(\log p)^2)$の計算量で計算できる計算であるが、測定をすると、平方根より時間のかかる計算であることがわった。

この結果をみると、最終的に１０進数ですべての桁数を出力する多倍長計算ではかなり多くの計算を行わない限り、10進数のままで計算した方が有利であることがわかる。計算

量が $O(p(\log p)^2)$ の円周率の計算ではこのことを考慮すると１０進数、実際には $10^4$ とか $10^8$ 進数で計算するのが有利であると言える。

## 参考文献


[1]右田、浅田、天野、藤野, 級数の再帰的集約による多倍長数の計算法とπの計算への応用, 情報処理学会論文誌, Vol.40, No.12, 1999, pp.4193-4200

[2]後 保範、金田 康正、高橋 大介, 級数に基づく多数桁計算の演算量削減を実現する分割有理数化法, 情報処理学会論文誌 ,Vol.41, No.6, 1998, pp.1811-1819

[3] 平山　弘, 分割統治法による多倍長演算の高速化,京都大学数理解析研究所講究録, Vol. 1138, 2000,pp.247-255

[4] Haible B., Papanikolaou T., Fast multiprecision evaluation of series of rational numbers, Technical Report No. TI-7/97, Darmstadt University of Technology, http://www.informatik.tu-darmstadt.de/TI/Mitarbeiter/papanik/

[5] 平山　弘, C++言語による高精度計算パッケージの開発,日本応用数理学会, Vol. 5, No.3, 1995, pp. 123-134

[6] Bergland G. D., A Radix-Eight Fast Fourier Transform Subroutine for Real-Valued Series, IEEE Trans. A. E., AU17.2,1969, pp. 138-144