数理解析研究所講究録 1456 巻 2005 年 174-179

指数対数関数等の超越関数の多倍精度計算

平山 弘 (Hiroshi Hirayama)
神奈川工科大学*

森川 敦司 (Atsushi Morikawa)
神奈川工科大学

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 億桁の計算が行われた。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 1998 年から 1999 年にかけて、右田等 [3][4] や後等 [6] 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 [1] 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。

BSA 法は、分割統治法の一種で、大きな問題を小さな問題に分割し高速化を図るものである。別の言い方をすれば、本方式は分割した問題をトーナメント形式に計算することで高速化を図るものである。

高精度数の乗算については高速フーリエ変換 (FFT) を使う高速な計算方法がよく知られている。FFT は約 1000 桁以上の数値の乗算において有効である。BSA 法は 1000 桁以下の数値の計算でも高速に計算できる上、原理が単純なので、様々な計算において容易に利用することが出来る。

本論文では、指数対数関数の高精度計算として Taylor 展開に BSA 法を使って高速化する方法提案する。約 1000 桁以下の精度の計算では、Taylor 展開を使った計算が Sasaki and Kanada[5] によって、様々な計算法を比較して最も高速であることが示されているので、計算時間が問題となるのは、1000 桁以上の精度の計算である。ここで提案した Taylor 展開に BSA 法を適用して高速化した方法と Sasaki and Kanda によって提案された方法を 1000 桁を超えた精度で比較し、その高速性を示した。

以下の計算では、高精度計算プログラムとして、C++言語で自作したものを利用した。10000 進数表現の浮動小数点演算ルーチンである。1600 桁を超えると自動的に FFT を利用して計算する。

2 BSA 法

BSA 法は FFT と同様な分割統治法の一種で、大きな問題を小さな問題に分割して計算する方法である。

*hirayama@sd.kanagawa-it.ac.jp

2.1 簡単な例

B.Haible and T.Papanikolaou の論文にもあるが、階乗の計算が非常に良い例である。次の計算を行う。

$$n! = \prod_{k=1}^{n} k = 1 \cdot 2 \cdot 3 \cdots n \tag{1}$$

n が小さい場合、前から順番に計算しても計算時間に大きな違いが生じることはないが、n が大きい場合、前から順序よく計算する方法は効率的ではない。n が偶数ならば、2個づつ組み合わせ

$$n! = (1 \cdot 2) \cdot (3 \cdot 4) \cdots ((n-1) \cdot n) \tag{2}$$

n が4の倍数ならば、さらに2個づつ組み合わせ

$$n! = \{(1 \cdot 2) \cdot (3 \cdot 4)\} \cdots \{[(n-3) \cdot (n-2)] \cdot [(n-1) \cdot n]\} \tag{3}$$

計算する。このように計算することで、通常の計算方法よりも計算桁数が小さいままで計算を行うことが出来るので、計算速度の向上が期待できる。これは2のべき乗の場合に限らず、一般のnの場合においても同様のことがいえる。

2.1.1 階乗計算例

10000! の計算を行う。この計算では、BSA 法を使うだけでなく、1600 桁以上の数値に対しては FFT を利用して乗算を行っている。計算結果は

$$10000! = 2.8462596809170545189064132121198688 90 \times 10^{35659} \tag{4}$$

35660 桁の数値となる。この計算には厳密に計算するため5万桁の浮動少数点演算ルーチンを使って計算した。計算時間は Pentium 4 3.06GHz で行うと以下のようになる。このとき使用したコンパイラーは Borland C++ 6.0 である。

計算方法	計算時間 (msec)
BSA	47
従来の方法	3578

このように、大きな n に対する階乗計算は非常に速く計算できる。

2.1.2 級数の計算

次の形で表現できる級数は、

$$S = \frac{1}{C_0}(A_0 + \frac{B_0}{C_1}(A_1 + \frac{B_1}{C_2}(A_2 + \frac{B_2}{C_3}(A_3 + \frac{B_3}{C_4}(A_4 + \cdots \tag{5}$$

次のように変形することができる。

$$S = \frac{1}{C_0}(A_0 + \frac{B_0}{C_1 C_2}(A_1 C_2 + A_2 B_1 + \frac{B_1 B_2}{C_3}(A_3 + \frac{B_3}{C_4}(A_4 + \cdots \tag{6}$$

これらの関係を行列を使って表すと、(5) 式の第 n 項までの値 S_n は

$$\begin{pmatrix} Q_n & P_n \\ 0 & R_n \end{pmatrix} = \begin{pmatrix} B_0 & A_0 \\ 0 & C_0 \end{pmatrix} \begin{pmatrix} B_1 & A_1 \\ 0 & C_1 \end{pmatrix} \begin{pmatrix} B_2 & A_2 \\ 0 & C_2 \end{pmatrix} \cdots \begin{pmatrix} B_n & A_n \\ 0 & C_n \end{pmatrix} \tag{7}$$

となる。ここで $S_n = P_n/R_n$ である。(6) 式は、右辺第 2 と第 3 の行列をかけ算したものに相当する。この計算も、階乗の計算と同じように、乗算を行うことによって高速に計算することができる。

これらの計算の各数値は短い桁数の数値であるから、非常に高速に計算できる。桁数が大きくなった場合、高速フーリエ変換を利用した乗算法を使うと、$O(n(\log n)^3)$ の計算量で計算できる。

2.1.3 指数関数の計算

指数関数の Taylor 展開式から、(5) 式に相当する式は

$$e^x = \frac{1}{1}(1+\frac{x}{1}(1+\frac{x}{2}(1+\frac{x}{3}(1+\frac{x}{4}(1+\cdots(1+\frac{x}{n}(1+\cdots \tag{8}$$

となる。これを行列の積で表現すると

$$\begin{pmatrix} Q_n & P_n \\ 0 & R_n \end{pmatrix} = \begin{pmatrix} x & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x & 1 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} x & 1 \\ 0 & 3 \end{pmatrix}\cdots\begin{pmatrix} x & 1 \\ 0 & n \end{pmatrix} \tag{9}$$

となる。

2.1.4 対数関数の計算

対数関数の場合も同様で Taylor 展開式から、(5) 式に相当する式は

$$\frac{\log(1+x)}{x} = \frac{1}{1}(1-\frac{x}{2}(1-\frac{2x}{3}(1-\frac{3x}{4}(1-\cdots(1-\frac{(n-1)x}{n}(1-\cdots \tag{10}$$

行列の乗算形式に変形すると

$$\begin{pmatrix} Q_n & P_n \\ 0 & R_n \end{pmatrix} = \begin{pmatrix} -x & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} -2x & 1 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} -3x & 1 \\ 0 & 3 \end{pmatrix}\cdots\begin{pmatrix} -nx & 1 \\ 0 & n \end{pmatrix} \tag{11}$$

このほか、三角関数、逆三角関数、双曲線関数など簡単な規則で各項の係数が表現でき、多くの関数がこの行列の乗算形式に変形できます。Taylor 展開の係数が簡単な規則で表現できない $\tan x$ が例外的に表現できないだけである。

2.2 連分数の計算法

級数と同様に、連分数の計算も BSA 法によって高速化することが出来る。(有限項で打ち切った連分数を次のように定義する。

$$\frac{P_n}{Q_n} = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{\ddots + \cfrac{a_n}{b_n}}}} = b_0 + \frac{a_1}{b_1} + \frac{a_2}{b_2} + \cdots + \frac{a_n}{b_n} \tag{12}$$

このとき、よく知られているように以下の漸化式が成り立つ。

$$\begin{aligned} P_n &= b_n P_{n-1} + a_n P_{n-2} \\ Q_n &= b_n Q_{n-1} + a_n Q_{n-2} \end{aligned} \qquad (n = 1, 2, \cdots) \tag{13}$$

だだし、$Q_0 = 1, P_0 = b_0, P_{-1} = 1, Q_{-1} = 0$ とする。この連分数の漸化式は次のような行列の乗算の形式で書くことができる。

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} P_{n-1} & P_{n-2} \\ Q_{n-1} & Q_{n-2} \end{pmatrix} \begin{pmatrix} b_n & 1 \\ a_n & 0 \end{pmatrix} \tag{14}$$

この関係式を利用すると、(12) 式は、

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 & 1 \\ a_0 & 0 \end{pmatrix} \begin{pmatrix} b_1 & 1 \\ a_1 & 0 \end{pmatrix} \begin{pmatrix} b_2 & 1 \\ a_2 & 0 \end{pmatrix} \cdots \begin{pmatrix} b_n & 1 \\ a_n & 0 \end{pmatrix} \tag{15}$$

となる。ここで、$a_0 = 1$ である。この式を使って、P_n/Q_n を計算すれば、連分数の値を計算できる。

2.2.1 逆正接関数の計算法

逆正接関数 $\tan x^{-1}$ は次のように連分数展開できる。

$$\tan^{-1} x = \frac{x}{1} + \frac{x^2}{3} + \frac{4x^2}{5} + \frac{9x^2}{7} + \cdots + \frac{n^2x^2}{2n+1} + \cdots \tag{16}$$

これを行列の乗算形式になおすと

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ x & 0 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ x^2 & 0 \end{pmatrix} \begin{pmatrix} 5 & 1 \\ 4x^2 & 0 \end{pmatrix} \cdots \begin{pmatrix} 2n+1 & 1 \\ n^2x^2 & 0 \end{pmatrix} \tag{17}$$

となる。$\pi = 4\tan^{-1} 1$ となるので $x = 1$ を代入することで計算できる。この式を使って 10 万桁計算するのに Athlon800MHz において 4 5 秒で計算できた。

$\tan^{-1} 1$ を計算するために、$\tan^{-1} x$ の Taylor 展開式に $x = 1$ を代入することは理論的には収束するが、非常に遅く、実用的には収束しない級数のため、このような方法でこの式の値を計算することには無理がある。しかし、$\tan^{-1} x$ に対応する連分数においては、実用に耐えうる速度で収束するため、このように計算すること可能である。

2.3 指数関数の高精度計算

指数関数は次の方法で計算した。

$$e^x = \left(e^{\frac{x}{2^n}}\right)^{2^n} \tag{18}$$

この式は、$t = \frac{x}{2^n}$ と置くと、次の 2 式に分割できる。

$$e^t = 1 + t + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots \tag{19}$$

と

$$e^x = \left(e^t\right)^{2^n} \tag{20}$$

に分割できる。(19) と (20) の計算量の総和が小さくなるように n を選ぶ。このとき、n は計算精度を 10 進での桁数 P とした時 $n = 2.25\sqrt{P}$ となる。ただし 2.25 の数値は数値実験によって決定した値である。

2.3.1 BSA 法の利用

e^t を高速に計算するために、次のように変形する。

$$e^t = e^{t_0+t_1} = e^{t_0}e^{t_1} \tag{21}$$

ここで t_0 に小さい桁の数をとり、t_1 にはその先の数値にとる。このように取ると、e^{t_0} は t_0 の桁数が少ないため BSA 法で高速化できる。e^{t_1} の値は t_1 がきわめて小さいため急速に収束する。このことから数値を少ない上位桁と残りの桁に分割することは計算する上で非常に有効な方法となる。

この方法で、5000 桁の e^{π} を計算した。

$$e^{\pi} = 23.140692632779269005729086367948547380266106$$

これまで 530msec かかったものを、BSA 法を使うことによって、293msec に短縮できた。

2.4 対数関数の高精度計算

対数関数は次の方法で計算した。

$$x = \log t = x_0 + \log\left(1 + \frac{t - t_0}{t_0}\right), x_0 = \log t_0 \tag{22}$$

ここで t_0 として小さい桁の近似値をとる。$\log t_0$ は t_0 の桁数が少ないため BSA 法で高速化できる。$\log\left(1 + \frac{t-t_0}{t_0}\right)$ の値は $\frac{t-t_0}{t_0}$ がきわめて小さいためすぐに収束する。

上の式を適用する前に、BSA 法の計算を速く収束させるために、

$$t = \frac{x}{2^n} \tag{23}$$

として、(22) で使う値を 1 以下になるように n を選んだ。このようにすると、$\log 2$ の定数値の高精度の値が必要となる。これを次の式を利用して、計算した。

$$\log(2) = 3\log(81/80)+5\log(25/24)+7\log(16/15)$$

log(81/80)、log(25/24)、log(16/15) は、BSA 法を使って、次の式

$$\log\frac{x+1}{x-1} = \log\frac{1+\frac{1}{x}}{1-\frac{1}{x}} = 2(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \frac{1}{7x^7} + \cdots \tag{24}$$

を使って計算した。

この方法で、5000 桁の $\log\pi$ を計算した。

$$\log\pi = 1.144729885849400174143427351353058711647294 8$$

従来の方法で 431msec かかったものを、BSA 法を使うことによって、375msec に短縮できた。これまでの対数の高速高精度計算法として、Sasaki and Kanada による計算法が知られている。この方法で上の計算を行うと 385msec であった。ここで使った Sasaki and Kanada の方法とは、元の Sasaki and Kanada の方法では、高精度数の乗算には FTT を使用していないがここでは、FFT を利用した乗算を使った。さらに引数を 1000 分の 1 以下にする条件を 10000 分の 1 にした。

Sasaki and Kanada の方法では、高精度で計算された $\log 2$ と円周率の定数が必要であるがこれは事前に計算されているとしている。このため上の計算時間にはこれら定数計算の時間を含まない。

BSA 法による計算でも高精度の $\log 2$ の定数が必要であるが、毎回計算を行っている。もし $\log 2$ の計算が必要ないならば、上の問題の計算時間はさらに速くなり 250msec となる。

3 まとめ

指数関数や対数関数のTaylor展開にBSA法を適用することによって、BSAを使わない従来の方法に比べ40％程度の高速化ができた。対数関数に対しては、5000桁程度の精度で最も高速な計算方法として知られたSasaki and Kanadaの方法を超えることを示した。

参考文献

[1] 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/(1997)

[2] 平山、連分数の多倍精度高速計算法、情報処理学会論文誌 Vol.41, No.SIG8, pp. 85–91(2000)

[3] 右田剛史、天野晃、浅田尚紀、藤野清次: 級数の集約による多倍長数の計算法とπの計算への応用、情報処理学会研究報告、vol.98, No. 74, pp. 31–36(1998)

[4] 右田剛史、天野晃、浅田尚紀、藤野清次: 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, pp. 4193–4200(1999)

[5] Sasaki T., Kanada Y.: Practically Fast Multiple-Precision Evaluation of LOG(X), Journal of Information Processing, Vol.4, No.4, pp. 247–250(1982)

[6] 後保範、金田康正、高橋大介、無限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、京都大学数理解析研究所講究録 Vol.1084, pp. 60–71 (1999)

[7] 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, pp. 1811–1819 (2000)