連分数の多倍長精度高速計算法

平山 弘†

本論文では，小さな整数から構成されている連分数の N 桁の計算を Haible ら (1997) によって報告されている binary splitting algorithm を使うと $O(N(\log N)^3)$ の時間で計算できる高速計算法を提案する．この計算法を使えば，計算時間は計算精度 N が大きい計算において，binary splitting algorithm を使った級数の計算法や相加相乗平均法（AGM）と同等の計算時間で行うことができる．計算精度 N が小さい場合でも，この方法は従来の計算方法より高速な計算法になる．この計算法の有効性を示すために，いくつかの数値例を示した．

Fast Multiple-precision Calculation Method for Continued Fraction

Hiroshi Hirayama†

In this paper, we propose a fast algorithm for N digits calculation of continued fraction consist of short integer by binary splitting algorithm reported by Haible, et al. (1997) in $O(N(\log N)^3)$ time. With this algorithm, calculation time become comparable to that of binary splitting algorithm for sum of series and AGM method in case of large N. This method also gives faster algorithm than traditional one in case of small N. Some numerical examples are given to demonstrate the effectiveness of this algorithm.

1. はじめに

級数の計算において，計算を有理数で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 binary splitting algorithm（以降 BSA 法と略す）として，Haible ら[3]によって 1997 年に整理され公表されている．この文献を読むと BSA 法は，1976 年に 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．

本論文では，級数に適用できる BSA 法は，連分数の計算にも適用でき，それを利用することによって，いろいろな連分数を計算時間が級数と同じ程度の時間 $O(N(\log N)^3)$ で高速に計算することができることを示す．

最近，右田ら[9),10)]や後ら[12)]によって，BSA 法の級数への適用の再現が行われている．この事実を見る限り BSA 法は少なくとも日本ではよく知られた計算法とはいえないので，以下にその概略を示す．

Haible らでは，次の形の無限級数

$$S = \sum_{k=0}^{\infty} \left(\frac{a_k}{b_k} \frac{\prod_{j=0}^{k} p_j}{\prod_{j=0}^{k} q_j} \right) \tag{1}$$

だけでなく，さらに一般的な無限級数

$$U = \sum_{k=0}^{\infty} \left(\frac{a_k}{b_k} \left(\sum_{j=0}^{\infty} \frac{c_j}{d_j} \right) \frac{\prod_{j=0}^{k} p_j}{\prod_{j=0}^{k} q_j} \right) \tag{2}$$

の高速化も容易に行えることを述べている．ここで，a_k，b_k，c_k，d_k，p_k，q_k は，$O(\log k)$ の桁数の整数で，通常 k の定数係数の多項式である．BSA 法は，l，m，n $(l < m < n)$ を正の整数としたとき，次のような手順になる．まず，

$$S_{l,m-1} = \sum_{k=l}^{m-1} \left(\frac{a_k}{b_k} \frac{\prod_{j=0}^{k} p_j}{\prod_{j=0}^{k} q_j} \right) \tag{3}$$

$$P_{l,m-1} = \prod_{j=l}^{m-1} p_j \tag{4}$$

$$Q_{l,m-1} = \prod_{j=l}^{m-1} q_j \tag{5}$$

† 神奈川工科大学
Kanagawa Institute of Technology

$$B_{l,m-1} = \prod_{j=l}^{m-1} b_j \tag{6}$$

と記する．これを使って

$$T_{l,m-1} = B_{l,m-1}Q_{l,m-1}S_{l,m-1} \tag{7}$$

を定義する．このとき，次のような漸化式

$$P_{l,n-1} = P_{l,m-1}P_{m,n-1} \tag{8}$$

$$Q_{l,n-1} = Q_{l,m-1}Q_{m,n-1} \tag{9}$$

$$B_{l,n-1} = B_{l,m-1}B_{m,n-1} \tag{10}$$

$$T_{l,n-1} = B_{m,n-1}Q_{m,n-1}T_{l,m-1} + B_{l,m-1}P_{l,m-1}T_{m,n-1} \tag{11}$$

が成り立つ．この関係を何回も利用して，$l=0$ から大きな n までの $Q_{0,n-1}$，$B_{0,n-1}$，$T_{0,n-1}$ の値を求め

$$S_{0,n-1} = \frac{T_{0,n-1}}{B_{0,n-1}Q_{0,n-1}} \tag{12}$$

として，S の近似値を計算することができる．さらに

$$U_{l,m-1} = \sum_{k=l}^{m-1}\left(\frac{a_k}{b_k}\Big(\sum_{j=l}^{k}\frac{c_j}{d_j}\Big)\frac{\prod_{j=0}^{k}p_j}{\prod_{j=0}^{k}q_j}\right) \tag{13}$$

$$D_{l,m-1} = \prod_{j=l}^{m-1} d_j \tag{14}$$

$$C_{l,m-1} = D_{l,m-1}\sum_{j=l}^{m-1}\frac{c_j}{d_j} \tag{15}$$

$$V_{l,m-1} = D_{l,m-1}B_{l,m-1}Q_{l,m-1}U_{l,m-1} \tag{16}$$

と記するとき，次の漸化式

$$D_{l,n-1} = D_{l,m-1}D_{m,n-1} \tag{17}$$

$$C_{l,n-1} = C_{l,m-1}D_{m,n-1} + C_{m,n-1}D_{l,m-1} \tag{18}$$

$$V_{l,n-1} = D_{m,n-1}B_{m,n-1}Q_{m,n-1}V_{l,m-1} + D_{m,n-1}C_{l,m-1}B_{l,m-1}P_{l,m-1}T_{m,n-1} \tag{19}$$

が成り立つ．この関係式から，前の S と同様に，$l=0$ から大きな n までの $D_{0,n-1}$，$Q_{0,n-1}$，$B_{0,n-1}$，$V_{0,n-1}$ を求める．この値から

$$U_{0,n-1} = \frac{V_{0,n-1}}{D_{0,n-1}B_{0,n-1}Q_{0,n-1}} \tag{20}$$

を計算し，U の近似値を計算することができる．この計算法は，式 (1) および (2) において，a_k，b_k，c_k，d_k，p_k，q_k が小さな整数で，級数の値を高精度で計算する場合に効果的である．

効率的である理由は 2 つある．第 1 番目の理由は，フル精度の長い桁数の数値が途中で出現しないため小さな桁数のままで計算を進めることができるためである．式 (15) は，除数を因数として持つ $D_{l,m-1}$ を掛けるため整数値になるので，この除算のために長い桁数の数値が現れることはない．第 2 番目の理由は，数値の桁数が大きくなった場合，大きな桁数の数値の乗算に FFT を用いた高速乗算法を利用できるためである．

この種の計算は，通常，多倍長数間の加減算と多倍長数と整数や浮動小数点数の乗除算を使って計算を高速化する．そのときの計算量は $O(N^2)$ である．これに対し，この BSA 法によって計算した場合，計算量は計算桁数 N が大きいとき $O(N(\log N)^3)$ と高速になる．

右田ら[9),10)]や後ら[12)]による有理数を使った高速計算法は，式 (3) の級数を計算するために，漸化式 (8) から (11) を利用して，式 (12) によって S を計算する方法に相当する．

右田らの計算方法は，級数 S が，次の形で表現できるとき，すなわち

$$S = \frac{1}{C_0}\Big(A_0 + \frac{B_0}{C_1}\Big(A_1 + \frac{B_1}{C_2}\Big(A_2 + \frac{B_2}{C_3}\Big(A_3 + \frac{B_3}{C_4}\Big(A_4 + \cdots + \tag{21}$$

という形式で表されるとき，次のように変形する[4)]ことができる．

$$S = \frac{1}{C_0}\Big(A_0 + \frac{B_0}{C_1C_2}\Big(A_1C_2 + A_2B_1 + \frac{B_1B_2}{C_3}\Big(A_3 + \frac{B_3}{C_4}\Big(A_4 + \cdots + \tag{22}$$

式 (22) に変形された後も，式 (21) の形式を保つことから，このような変形は，式の途中で，何回でも行うことができる．式 (21) の A_i，B_i，C_i が小さな整数ならば，隣どうしをこのような変形を高速に行えるので，結果として，式 (21) が高速に評価できることになる．途中の整数の桁数が増えた場合，FFT を使って高速に計算できる．

また，この級数の第 n 項までの和の計算は，行列の次の式の乗算と同じ計算となる．

$$\begin{pmatrix} Q_n & P_n \\ 0 & R_n \end{pmatrix} = \prod_{k=0}^{n}\begin{pmatrix} B_k & A_k \\ 0 & C_k \end{pmatrix} \tag{23}$$

級数の値は，P_n/R_n となる．この行列は，隣り合う 2 つの行列を次々とトーナメント法的に乗算すれば，これまでの計算と同じになり高速に計算できる．

これまでの円周率の計算等でよく使われてきた算術幾何平均を使った方法は FFT を使った乗算の高速性をだけを利用しているため，FFT が効率的でない小さな桁数では従来の計算方法が有利になる．それに対し，BSA 法は小さな整数でできるだけ計算し，大きな

整数を扱わないことによって高速化を行っている．このため，FFT を利用した乗算法が効率的でない 10 進数 1000 桁程度以下の比較的精度の低い場合でも，高速に計算できる特徴がある．

FFT を使った乗算が有利になる桁数は，コンピュータに依存し，スカラ計算機では 10 進数で約 300 桁から約 10000 桁である．性能の高くない計算機では小さい桁数で FFT による乗算が有利になり，高性能計算機では高い桁数のところまで，従来の計算方法が有利になる傾向がある．300 桁は 10 MHz の Intel286+287（MS-FORTRAN）を使ったときの値であり，10000 桁は Pentium II 450 MHz で従来の乗算法をアセンブラで記述した場合である．Fortran や C++言語等の高級言語で記述した場合，最近のマイクロプロセッサでは，1000 桁から 2000 桁の間である．ベクトル・コンピュータの場合積和演算が高速になるように設計されているので，さらに高い桁数になる可能性がある．

2. 連分数の高速計算法

級数と同様な計算方法を，連分数に対しても行うことができる．この方法は，桁数の長い実定数を高速に連分数展開する Lehmer の方法[7]の逆計算に相当するものである．1938 年の論文であることから分かるように，この連分数展開法は筆算でできるような精度の計算でも効率的な計算法である．

有限項で打ち切った連分数を次のように記述する．ここで，a_k，b_k は，$O(\log k)$ の桁数の整数であるとする．a_k，b_k が k の整数係数の多項式である場合，この条件を満たす．本論文で扱うすべての連分数の係数がこの条件を満たしており，数値計算等でよく使われる連分数展開式もこの条件を満たしている．

$$\begin{aligned}\frac{P_n}{Q_n} &= b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{\ddots + \cfrac{a_n}{b_n}}}} \\ &= b_0 + \frac{a_1}{b_1} \;+\; \frac{a_2}{b_2} \;+\; \cdots \;+\; \frac{a_n}{b_n} \qquad (24)\end{aligned}$$

これらの連分数は，よく知られているように，次のような漸化式[8]によって計算できる．$P_{-1} = 1$，$Q_{-1} = 0$，$P_0 = b_0$，$Q_0 = 1$ として

$$\begin{aligned}P_n &= b_n P_{n-1} + a_n P_{n-2} \\ Q_n &= b_n Q_{n-1} + a_n Q_{n-2}\end{aligned} \qquad (25)$$

また，n から m までの部分連分数を次のように定義する．

$$\frac{P_{n,m}}{Q_{n,m}} = \frac{a_n}{b_n} \;+\; \frac{a_{n+1}}{b_{n+1}} \;+\; \cdots \;+\; \frac{a_m}{b_m} \qquad (26)$$

このように定義すれば，$P_n = P_{0,n}$，$Q_n = Q_{0,n}$ である．この定義より

$$\frac{P_{l,m}}{Q_{l,m}} = \frac{a_l}{b_l} \;+\; \cdots \;+\; \frac{a_{n-1}}{b_{n-1}} \;+\; \frac{P_{n,m}}{Q_{n,m}} \qquad (27)$$

が成り立つ．この式 (27) に関係式 (25) を適用すると，

$$\begin{aligned}P_{l,m} &= Q_{n,m}P_{l,n-1} + P_{n,m}P_{l,n-2} \\ Q_{l,m} &= Q_{n,m}Q_{l,n-1} + P_{n,m}Q_{l,n-2}\end{aligned} \qquad (28)$$

が得られる．この関係式は，m を $m-1$ としても成り立つ．

$$\begin{aligned}P_{l,m-1} &= Q_{n,m-1}P_{l,n-1} + P_{n,m-1}P_{l,n-2} \\ Q_{l,m-1} &= Q_{n,m-1}Q_{l,n-1} \\ &\quad + P_{n,m-1}Q_{l,n-2}\end{aligned} \qquad (29)$$

式 (28)，(29) を何回も適用することによって，計算を進めることができる．具体的な計算法としては，2 項を組み合わせると便利である．最初の 1 組の分数を

$$\begin{aligned}\frac{P_0}{Q_0} &= \frac{P_{0,0}}{Q_{0,0}} = \frac{b_0}{1} \\ \frac{P_1}{Q_1} &= \frac{P_{0,1}}{Q_{0,1}} = \frac{b_0 b_1 + a_1}{b_1}\end{aligned} \qquad (30)$$

と置く．次に

$$\begin{aligned}\frac{P_{n,n}}{Q_{n,n}} &= \frac{a_n}{b_n} \\ \frac{P_{n,n+1}}{Q_{n,n+1}} &= \frac{a_n}{b_n} \;+\; \frac{a_{n+1}}{b_{n+1}}\end{aligned} \qquad (31)$$

を計算する．式 (30) と (31) の値を使い，式 (28)，(29) を使って，4 項をまとめた式，さらに 8 項をまとめた式と次々計算する．最後に，1 つの分数の組にまとめる．この分数の分子を分母で除算すれば，この連分数の値が計算される．式 (30)，(31) の計算および式 (28)，(29) による項をまとめる計算は，独立に計算できるので，並列にも計算することができる．

最初に各項 2 項ずつまとめたが，計算対象の数値の桁数が小さく，計算機に備わっている浮動小数点数や整数 1 個を使って計算を進めることができる場合，さらに多くの項をまとめた方が効率的であると考えられる．

計算する級数の項数は，ソフトウエア的に指数部を拡張した倍精度浮動小数点[6]を使って連分数の値を計算し，収束条件を満たす項数を求める．収束判定には，関係式

$$\frac{P_n}{Q_n} - \frac{P_{n-1}}{Q_{n-1}} = (-1)^{n+1}\frac{a_1 a_2 a_3 \cdots a_n}{Q_n Q_{n-1}} \quad (32)$$

を利用する．関係式 (25) を利用して，Q_n だけを計算し，式 (32) の分子を別に計算することによって，計算量を少なくすることができる．

この計算では，指数部は大きくなるが，高精度計算は不要なので，高速に計算できる．

3. 行列を使った表現

右田らの方法と同様に，この計算式は行列を使って簡単に表現できる．漸化式 (25) は，

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} P_{n-1} & P_{n-2} \\ Q_{n-1} & Q_{n-2} \end{pmatrix} \times \begin{pmatrix} b_n & 1 \\ a_n & 0 \end{pmatrix} \quad (33)$$

と書くことができる．$P_{-1} = 1$，$Q_{-1} = 0$，$P_0 = b_0$，$Q_0 = 1$ の関係式を利用すると

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 & 1 \\ 1 & 0 \end{pmatrix} \times \prod_{i=1}^{n} \begin{pmatrix} b_i & 1 \\ a_i & 0 \end{pmatrix} \quad (34)$$

と表現できる．この行列を乗算し，P_n，Q_n を求め，P_n/Q_n を計算することによって連分数の値を計算することになる．

高速計算方法は，隣り合う行列をトーナメント的に，次々乗算する方法に対応する．右田らの方法と同じように 2 行 2 列の行列の積で表現できるが，計算を進めると行列要素はすべて 0 でない値が入るため，計算項数が同じなら，つねに上三角行列となる右田らの方法がより短い計算時間で計算できる．

4. 計 算 量

N 桁の数の乗算の計算量を $M(N)$ とする．$M(N)$ は，通常 $O(M(N)) = O(N^2)$ であるが，N が大きな数のとき，高速フーリエ変換（FFT）[1),2),11)]を利用して，$O(M(N)) = O(N \log N)$ で計算できることが知られている．加減算の計算量は，$O(N)$，除算の計算量は，$O(M(N))$ である．加減算は，乗算に比べ計算量が少ないのでここでは無視する．除算は 1 回だけなので無視し，乗算の回数で計算量を推定する．

a_k，b_k の最大桁数を K，計算する項数を L $(= 2^n)$ とする．式 (28)，(29) を 1 回計算するのに 8 回の乗算が必要だから，$L/2$ 回の計算を必要とするので，合計 $4L$ 回の K 桁の数値の乗算を必要とする．計算量は $4LM(K)$ となる．次の段階では，式 (27) を使うと，項数は $L/4$，計算量は $2LM(2K)$ となる．さらに，次の段階では，計算桁数は倍，計算項数は半分になるので，$LM(4K)$ となる．したがって，計算量は，$O(M(N)) = O(N \log N))$ であることを考慮し，計算すべき項数 L と計算桁数 N には，ほぼ $L = kN$（k は定数）が成り立ち，さらに，$K = l \log N$（l は定数）が成り立つと仮定すると，

$$\begin{aligned} &\sum_{i=0}^{n-1} 2^{2-i} LM(2^i K) \\ &\quad \approx L\sum_{i=0}^{n-1} 2^{2-i} O(2^i K(\log_2 K + i)) \\ &\quad \approx 4L\sum_{i=0}^{n-1} O(K(\log_2 K + i)) \\ &\quad \approx 4LKO(n(\log_2 K) + \frac{n(n-1)}{2}) \\ &\quad \approx O(2LKn(n-1)) \\ &\quad \approx O(N(\log N)^3) \end{aligned} \quad (35)$$

が得られる．計算量は，$O(N(\log N)^3)$ となる．この計算量の評価で注意すべきことは，計算桁数 N が，多くの計算機では 10 進数で 1000 桁以上の大きいときの評価であることである．計算する桁数が小さい場合，FFT を使った乗算より通常の乗算法が高速になるのは，計算量が FFT を使った評価より小さくなるためである．

5. 連分数よる計算例

連分数を BSA 法で計算した場合の具体例を示す．時間測定に使用したコンピュータは，メモリ 512 M バイト Pentium III 733 MHz である．コンパイラとして，C++Builder5 を使用した．

5.1 逆正接関数の計算

連分数展開式は，一般に，べき級数展開式に比較し，収束範囲が広い，収束が速い等の性質があることが知られている．たとえば，$\tan^{-1} x$ のべき級数展開式は

$$\tan^{-1} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots \quad (36)$$

と表現できる．また，連分数展開すると

$$\tan^{-1} x = \frac{x}{1} + \frac{x^2}{3} + \frac{4x^2}{5} + \frac{9x^2}{7} + \cdots + \frac{n^2 x^2}{2n+1} + \cdots \quad (37)$$

となる．行列表現すると

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ x & 0 \end{pmatrix} \times \prod_{i=1}^{n} \begin{pmatrix} 2i+1 & 1 \\ i^2x^2 & 0 \end{pmatrix} \quad (38)$$

となる．これらの式で，$x = 1$ として $\pi/4$ を 1 万桁の精度で計算するとすると，式 (36) では，項数を多くとらなければならない．1000 万項まで計算しても 7 桁の精度しか得られないため，1 万桁の計算は実際上不可能である．連分数展開式（式 (38)）では，円周率の計算としては効率的ではないが，計算することが可能であり，次のように計算することができる．

BSA 法を使うと，1 万桁を 1.602 秒で計算することができる．式 (24) の漸化式を用いた通常の計算法の場合でも 47.706 秒である．BSA 法は，式 (24) の漸化式を用いた通常の計算法に比べ約 30 倍高速である．1 万桁の円周率の計算の場合，分子分母ともに約 53000 桁の整数になる．この分数の計算は，要求精度が 1 万桁であるので，仮数部 1 万桁の浮動小数点演算を行うことで高速化を実現している．FFT が効率的でない 1000 桁の計算の場合，BSA 法では，0.06 秒で計算が終了し，通常法の 0.451 秒に比べ約 7.5 倍の高速化が達成されている．BSA 法を用いた円周率 10 万桁の計算では 130626 項までの計算が必要であり，44.584 秒で計算が終了する．

逆正接関数の計算は，べき級数展開で計算しようとすると収束が遅かったり，または発散してしまうが，連分数展開で計算するとべき級数展開の場合にくらべ収束が速くなり，実用的な時間で時間で計算を実行することができる例の 1 つである．

5.2 巡回連分数の計算

整数の平方根は，巡回連分数に展開できる．この性質を使うと，1 周期の部分を部分連分数として計算すれば，式 (28)，(29) により，2 周期の部分連分数が計算できる．さらに 4 周期，8 周期と次々と計算できる．このように周期を単位に分割すると，すべての部分連分数が同じになるため，計算量を大幅に減らすことができる．ここでは，具体例として，$\sqrt{2}$ の値を計算する．$\sqrt{2}$ は，次ように連分数展開される．

$$\sqrt{2} = 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \cdots \quad (39)$$

これは，次のようなに行列表現ができる．

$$\begin{pmatrix} P_n & P_{n-1} \\ Q_n & Q_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix}^n \quad (40)$$

この計算の場合，最初の定数 1 を除けば，途中の部分連分数は，すべて同じになるので，計算を簡略化することができ，効率的に計算できる．行列の表現でいえば，行列の単なる積ではなく，べき乗の計算になるのでさらに効率的に計算できる．この計算法で計算すると，1 万桁の計算で 0.09 秒であった．連分数の漸化式を通常どおり計算すると 7.05 秒で約 78 倍の差となる．

5.3 いろいろな関数値の計算

対数関数も次の公式を利用して計算できる．

$$\log\left(\frac{1+x}{1-x}\right) = \frac{2x}{1} - \frac{x^2}{3} - \frac{4x^2}{5} - \frac{9x^2}{7} - \cdots - \frac{n^2x^2}{2n+1} - \cdots \quad (41)$$

この式に $x = 1/3$ を代入すると，$\log 2$ の値を計算できる．連分数の BSA 法で計算すると 1 万桁を 0.761 秒で計算することができた．通常の漸化式を利用して計算すると，18.626 秒で約 24 倍の差である．

上に示した式以外にも，e^x，$\sin^{-1} x$，$\tan x$ 等[8]の展開式がある．$\tan x$ はべき級数展開式と異なり，次のような簡単な式になる．

$$\tan x = \frac{x}{1} - \frac{x^2}{3} - \frac{x^2}{5} - \frac{x^2}{7} - \cdots - \frac{x^2}{2n+1} - \cdots \quad (42)$$

このように係数が容易に推定できる形に展開できる．逆に，$\sin x$，$\cos x$ 等の連分数展開式は，係数が簡単な規則で得られる形には展開できない．

e^x は，連分数展開式でも簡単に書けて

$$e^x = \frac{1}{1} - \frac{x}{1} + \frac{x}{2} - \frac{x}{3} + \frac{x}{2} - \frac{x}{5} + \cdots + \frac{x}{2} - \frac{x}{2n+1} + \cdots \quad (43)$$

となる．また，連分数の縮約によって

$$e^x = 1 + \frac{2x}{2-x} + \frac{x^2}{6} + \frac{x^2}{10} + \frac{x^2}{14} + \cdots + \frac{x^2}{4n+2} + \cdots \quad (44)$$

と書くこともできる．

6. 連分数による円周率の計算

円周率の計算法としては，最近使われなくなったが，よく引用されるので比較のため Machin の公式による円周率の計算を行った．また，連分数による比較的効率的な計算法を紹介する．

6.1 Machin の公式の計算

他の公式との比較のために，円周率の計算によく使われる Machin の公式による計算を行う．

$$\pi = 16\tan^{-1}\frac{1}{5} - 4\tan^{-1}\frac{1}{239} \tag{45}$$

この公式を使うために，式 (37) を次のように変形する．

$$\tan^{-1}\frac{1}{p} = \frac{1}{p} + \frac{1}{3p} + \frac{4}{5p} + \cdots + \frac{n^2}{(2n+1)p} + \cdots \tag{46}$$

関係式 (45) と連分数展開式 (46) を利用し，BSA 法を適用すれば，円周率を計算することができる．5000 桁で 0.33 秒，1 万桁で 0.84 秒，2 万桁で 2.24 秒，3 万桁で 3.62 秒，4 万桁で 5.57 秒，5 万桁で 7.41 秒，6 万桁で 9.81 秒で計算できる．これらの結果は，ほぼ $O(N(\log N)^3)$ の結果となり，理論と一致している．

さらに高精度の計算を行うと，8 万桁で 16.54 秒，10 万桁で 23.67 秒，20 万桁で 163.98 秒，50 万桁で 647.8 秒，100 万桁で 2124.4 秒で計算できる．

この計算時間から計算桁数が大きくなると計算時間が理論以上に増加している．これは，この計算には FFT を利用した C++言語の汎用多倍長演算ルーチン[5),6)]を使用しているため，桁数の多い一時変数の割付け，解除が頻繁に発生するためだと思われる．このため，この実行プログラムのユーザ作成部分ではできるだけ一時変数の割付け，解除を減らすようにプログラムを書き，計算時間を短縮を図っている．この効果は，計算桁数が大きいときほど効果的である．

6.2 Ramanujan の gamma 関数の関係式

連分数による円周率を効率的に計算する公式はあまり知られていない．ここでは，Ramanujan の gamma 関数の関係式[8)]

$$\frac{z}{4}\frac{\Gamma^2(\frac{z}{4})}{\Gamma^2(\frac{z+2}{4})} = 1 + \frac{2}{2z-1} + \frac{1\cdot 3}{2z} + \frac{3\cdot 5}{2z} + \cdots + \frac{(2n+1)\cdot(2n+3)}{2z}\cdots \tag{47}$$

を使った計算法を紹介する．この式に $z = 4n$ を代入し，$\Gamma(1/2) = \sqrt{\pi}$ を考慮して，円周率について解くと

$$\pi = \frac{1}{n}\left(\frac{2\cdot 4\cdot 6\cdots(2n)}{1\cdot 3\cdot 5\cdots(2n-1)}\right)^2 \times \left(1 + \frac{2}{8n-1} + \frac{1\cdot 3}{8n} + \frac{3\cdot 5}{8n} + \cdots\right) \tag{48}$$

が得られる．この公式は，他の円周率の計算公式と異なり，任意定数 n を含む．この定数を適当に設定することによって，効率的な計算を行うことができる．

この計算を実行する前に，数十万桁の計算を何回か行い，計算時間が最小となる n の値を調べた．その結果，計算したい桁数と同程度（0.7〜1.0）の数値にするとほぼ最良の結果が得られることが分かった．100 万桁の計算ならば，n は 700000 から 1000000 程度の数値を与える．この範囲の数値ならばそれほど計算時間に大きな差が生じることはない．以下の計算では n は計算桁数の 8 割の値に設定して計算した結果である．

1 万桁で 0.73 秒，2 万桁で 1.79 秒，3 万桁で 2.92 秒，4 万桁で 4.49 秒，5 万桁で 5.61 秒，6 万桁で 6.99 秒，10 万桁で 15.25 秒，20 万桁で 54.58 秒，50 万桁で 181.9 秒，100 万桁で 497.5 秒で計算できる．この計算時間から，前節の Machin の公式より速く計算できることが分かる．Machin の公式の場合と同様に 6 万桁以下では，計算時間は $O(N(\log N)^3)$ になっている．

この公式では n を大きくとれば，収束がそれだけ速くなるので，高精度の円周率の計算において，これまで知られている公式より高速に計算できる可能性がある．

これまで連分数の高速な計算法が知られていなかったため，連分数展開を利用した円周率計算公式の研究が行われてこなかった．このため，連分数を使った効率の良い円周率計算公式はあまり知られていない．連分数の高速計算法の研究が進めば，さらに効率の良い計算公式が開発され，高速に計算できるようになると思われる．

7. ま と め

計算精度に比べ小さな桁数の数値で構成された連分数の N 桁の計算は，通常 $O(N^2)$ の演算量を必要とするが，本論文では BSA 法を用いて $O(N(\log N)^3)$ で可能であることを示した．この計算時間は，これまで知られている高速級数計算法や相加相乗平均を利用した計算方法とほぼ同じオーダである．

この方法は，計算桁数が多いときに高速であるだけでなく，FFT による乗算法が効率的でない 10 進数で

数百桁の計算でも効率的である．連分数の収束領域はべき級数と比較し広いことが知られているので，べき級数では収束が遅く計算が難しい関数も効率的に計算できるようになると期待できる．

参 考 文 献

1) Bergland, G.D.: A Radix-Eight Fast Fourier Transform Subroutine for Real-Valued Series, *IEEE Trans. A. E.*, AU17.2, pp.138–144 (1969).
2) Henrici, P.: *Applied and Computational Complex Analysis*, Vol.3, Chap.13, John Wiley & Sons, New York (1986).
3) Haible, B. and Papanikolaou, T.: Fast multiprecision evaluation of series of rational numbers, Technical Report, No.TI-7/97, Darmstadt University of Technology (1997).
4) 平山 弘，浮川直章：連分数の高速評価法，情報処理学会研究報告，Vol.99, No.74, pp.31–36 (1998).
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, pp.123–134 (1995).
6) 平山 弘：多倍長計算プログラムパッケージ MPPACK の利用の手引き，東大計算センター (1992).
7) Lehmer, D.H.: Euclid's Algorithm for Large Numbers, *Amer. Math. Monthly*, Vol.45, pp.227–233 (1938).
8) Lorentzen, L. and Waadeland, H.: *Continued Fractions with Applications*, North-Holland, Amsterdam (1992).
9) 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).
10) 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).
11) 森，名取，鳥居：数値計算（岩波講座情報科学 18），5 章，岩波書店 (1982).
12) 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).

(平成 12 年 5 月 9 日受付)
(平成 12 年 9 月 5 日採録)

平山 弘（正会員）

1952 年生．1980 年東京大学大学院理学系研究科博士課程物理学専攻修了．理学博士．神奈川工科大学助教授．数学ソフトウエア，高精度計算，C++言語の応用に関する研究に従事．日本物理学会，日本流体力学会，日本機械学会，日本応用数理学会，日本数式処理学会各会員．