アルゴリズム 5－3
（1989. 1. 25）

連分数を用いた平方根の高速発生法とその計算効率

小沢　一文　海野　啓明
（仙台電波工業高等専門学校）

本報告では、整数nの平方根にr次（$r>1$）収束するような有理近似列を生成するアルゴリズムを導出し、その計算効率について考察する。このアルゴリズムは、平方根の連分数近似を加速したものであるため、生成される有理数列は常に既約である。また、有理近似の分母と分子を計算する過程では除算を含まないため、丸め誤差の影響を全く考慮する必要がない。計算効率を詳細に調べた結果、5次収束法が最適であることが判明した。

A Fast Algorithm for Calculating Continued Fractions of Square Root and It's Complexity

Kazufumi Ozawa and Keimei Kaino
Sendai National College of Technology,
Kami-Ayashi Aza Kitahara 1 Sendai, 989-31, Japan.

Abstract

In this report an r'th order converging algorithm is derived for approximating square root of integers, and the complexity of the algorithm is considered. Since the algorithm is an accelerated version of continued fraction expansion, the sequence generated by this algorithm is always irreducible. Moreover, since the algorithm does not contain divisions during the calculations of the denominator and the numerator, this algorithm is free from round-off error. The complexity analysis of the algorithm and the numerical experiments show that the algorithm is best for r=5.

1．はじめに

整数nが非平方数であるとき、$\sqrt{n}$は無理数になることが知られている。しかし、実用上はこれを有理数で近似する必要がある。通常の浮動小数点演算ルーチンでは、平方根の有理近似法として、Newton法を

$$x_{k+1} = \frac{1}{2}\left(x_k + n/x_k\right) \tag{1.1}$$

という形で用いている。このアルゴリズムは、よく知られているように2次収束法なので、充分に良い初期値を選べば、通常の計算では一回の反復で所望の精度が得られるようになっている。しかし除算を含むため、多倍長演算を用いて平方根を超高精度で求めるときには、あまり適切なアルゴリズムだとはいえない。そこで、著者等は平方根$\sqrt{n}$の有理近似の分子、分母を並行に計算し、希望の精度が得られたところで初めて除算を行うようなアルゴリズムを提案する。このアルゴリズムは、平方根の連分数展開を加速したものであり、収束の次数を任意に高くできる。また、ここでは多倍長演算を用いることを前提として計算効率を考察し、この任意次数の収束性を持つアルゴリズム群のなかで、計算効率の最も高いものを決定する。

2．平方根の連分数近似

整数$n > 0$が非平方数のとき$\sqrt{n}$は無理数になり、以下に示すような周期Nの連分数に展開できることが知られている[1]：

$$\omega = \sqrt{n} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + a_N}}} \tag{2.1}$$

ここで、各a_kは

$$a_1 = a_{N-1},\quad a_2 = a_{N-2},\quad \cdots,\quad a_N = 2a_0 \tag{2.2}$$

といった周期性を持っている[2]。$a_0, a_1, \cdots, a_k$からなるωの中間近似分数をP_{k+1}/Q_{k+1}で表すと、P_{k+1} Q_{k+1}は漸化式

$$P_{k+1} = a_k P_k + P_{k-1}, \tag{2.3}$$

$$Q_{k+1} = a_k Q_k + Q_{k-1}, \tag{2.4}$$

$$k = 0, 1, 2, \ldots,$$

$$P_{-1} = 0,\quad P_0 = 1,$$

$$Q_{-1} = 1,\quad Q_0 = 0,$$

を満たす。これを行列表示すると、

$$\begin{bmatrix} P_{k+1} & Q_{k+1} \\ P_k & Q_k \end{bmatrix} = \begin{bmatrix} a_k & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P_k & Q_k \\ P_{k-1} & Q_{k-1} \end{bmatrix} \tag{2.5}$$

となる。これより、P_N, Q_Nは

$$\begin{bmatrix} P_{N+1} & Q_{N+1} \\ P_N & Q_N \end{bmatrix} = H(a_N) \cdots H(a_0) = L \cdot H(a_0) \tag{2.6}$$

を満たすことがわかる。ここで、

$$H(a)=\begin{bmatrix} a & 1 \\ 1 & 0 \end{bmatrix},\quad L=H(a_N)\cdots H(a_0) \tag{2.7}$$

と置いた。

ここで、数列$\{P_k\}$, $\{Q_k\}$の一周期毎の値P_{mN}, Q_{mN}をそれぞれU_m, V_mで表すと、

$$U_m=\frac{1}{2}(\lambda_1^m+\lambda_2^m) \tag{2.8}$$

$$V_m=\frac{1}{2}(\lambda_1^m-\lambda_2^m) \tag{2.9}$$

$$m=0,1,2,\ldots.$$

となる[3]。ただし、λ_1, λ_2は行列Lの固有値で条件

$$\lambda_1>1,\quad |\lambda_2|<1 \tag{2.10}$$

を満たしている。これらの結果より、U_m/V_mの持つ誤差は十分大きなmに対して

$$\frac{U_m}{V_m}-\omega=\frac{2\lambda_2^m}{\lambda_1^m-\lambda_2^m}$$

$$\simeq 2\omega(\lambda_2^2)^m \tag{2.11}$$

と近似できることがわかる。また、

$$|U_m/V_m-\omega|\leqq(1/2\omega)/V_m^2 \tag{2.12}$$

という評価もできる。式(2.11)は、数列$\{U_m/V_m\}$が$\omega=\sqrt{n}$に漸近的に一次収束することを示している。

3．r次収束列について

つぎに、$\sqrt{n}$にr（>1）次収束する有理数列を生成するアルゴリズムを提案する。まず、U(x, y), V(x, y)なる関数を次のように定義する：

$$U(x,y)=\frac{1}{2}(x+y) \tag{3.1}$$

$$V(x,y)=\frac{1}{2\omega}(x-y) \tag{3.2}$$

これより、x, yは

$$x=U+\omega V \tag{3.3}$$

$$y=U-\omega V \tag{3.4}$$

として表せるから、

$$U(x^r,y^r)=\{(U+\omega V)^r+(U-\omega V)^r\}/2$$

$$=\sum_{i=0}^{\lfloor r/2\rfloor}\binom{r}{2i}U(x,y)^{r-2i}V(x,y)^{2i}n^i, \tag{3.5}$$

$$V(x^r,y^r)=\{(U+\omega V)^r-(U-\omega V)^r\}/2\omega$$

$$=\sum_{i=0}^{\lfloor \frac{r-1}{2}\rfloor}\binom{r}{2i+1}U(x,y)^{r-2i-1}V(x,y)^{2i+1}n^i, \tag{3.6}$$

なる2つの恒等式が得られる。ここで、$x=\lambda_1^m$, $y=\lambda_2^m$と置けば、

$$U_{rm}=U\left(\lambda_1^{rm},\ \lambda_2^{rm}\right)=\sum_{i=0}^{\lfloor r/2\rfloor}\binom{r}{2i}U_m^{r-2i}V_m^{2i}n^i, \tag{3.7}$$

$$V_{rm}=V\left(\lambda_1^{rm},\ \lambda_2^{rm}\right)=\sum_{i=0}^{\lfloor (r-1)/2\rfloor}\binom{r}{2i+1}U_m^{r-2i-1}V_m^{2i+1}n^i \tag{3.8}$$

が得られる。式(3.7),(3.8)は、連分数近似の第mステップの値U_m, V_mに対して簡単な多項式演算を施せば、第rmステップの値U_{rm}, V_{rm}が作りだせることを意味している。また、連分数の中間近似分数列は常に既約な分数列であるので[2]、式(3.7),(3.8)を反復することによって既約な分数列を高速に作りだせる。この数列はr次収束列であることが次の定理によって示される：

〔定理〕

漸化式

$$U_{k+1}^{(r)}=\sum_{i=0}^{\lfloor r/2\rfloor}\binom{r}{2i}\left(U_k\right)^{r-2i}\left(V_k^{(r)}\right)^{2i}n^i, \tag{3.9}$$

$$V_{k+1}^{(r)}=\sum_{i=0}^{\lfloor (r-1)/2\rfloor}\binom{r}{2i+1}\left(U_k\right)^{r-2i-1}\left(V_k^{(r)}\right)^{2i+1}n^i, \tag{3.10}$$

$$k=0,1,\ldots$$

$$U_0^{(r)}=U_1=P_N,\quad V_0^{(r)}=V_1=Q_N$$

によって生成される数列は$\sqrt{n}$にr次収束する。

〔証明〕式(3.7),(3.8)より、$U_k^{(r)}=U_{r^k}$, $V_k^{(r)}=V_{r^k}$であるから、式(2.11)より

$$e_k^{(r)}\equiv\left(U_k^{(r)}/V_k^{(r)}\right)-\omega=2\left(\lambda_2\right)^{r^k}/\left[\left(\lambda_1\right)^{r^k}-\left(\lambda_2\right)^{r^k}\right]\cdot\omega$$

を得る。これより、

$$\lim_{k\to\infty}\frac{e_{k+1}^{(r)}}{\left(e_k^{(r)}\right)^r}=(2\omega)^{1-r} \tag{3.11}$$

となって、アルゴリズム(3.9),(3.10)のr次収束性は示された。 Q.E.D.

以下いくつかのrについてr次収束法の具体例を示す。

(i) r = 2

この場合は、式(3.9),(3.10)より

$$U_{k+1}^{(2)}=\left(U_k^{(2)}\right)^2+n\left(V_k^{(2)}\right)^2,$$

$$V_{k+1}^{(2)}=2U_k^{(2)}\cdot V_k^{(2)}, \tag{3.12}$$

$$k=0,1,2,\ldots.$$

を得る。この式において$x_k=U_k/V_k$と置けば、アルゴリズム(3.12)はNewton法(1.1)になることが容易にわかる。ここで、式(2.8)、(2.9)から得られる恒等式 (Pell方程式)[4]

$$n\left(V_k^{(r)}\right)^2=\left(U_k^{(r)}\right)^2-\left((-1)^N\right)^{r^k} \tag{3.13}$$

$$k=0,1,2,\ldots\ldots,$$

を、式(3.12)に代入すると、

$$U_{k+1}^{(2)}=2\left(U_k^{(2)}\right)^2-1,$$

$$V_{k+1}^{(2)}=2U_k^{(2)}\cdot V_k^{(2)}, \tag{3.14}$$

となりアルゴリズムは簡略化される。J.Dutka[5]はこれとほとんど等価な式を用いて $\sqrt{2}$ を100万桁計算した。

(ii) r = 3

式(3.10),(3.11)で、r = 3 と置くと

$$U_{k+1}^{(3)} = \left(U_k^{(3)}\right)^3 + 3\left(U_k^{(3)}\right)\left(V_k^{(3)}\right)^2 n,$$
$$V_{k+1}^{(3)} = 3\left(U_k^{(3)}\right)^2\left(V_k^{(3)}\right) + \left(V_k^{(3)}\right)^3 n, \qquad (3.15)$$

を得る。このアルゴリズムは、方程式

$$x^2 - n = 0$$

にBailey法[6]を適用したものと等価になる。ここで恒等式(3.13)を用いて簡略化すると次のようになる：

$$U_{k+1}^{(3)} = U_k^{(3)} \cdot \{4\left(U_k^{(3)}\right)^2 - 3\ (-1)^N\}$$
$$U_{k+1}^{(3)} = V_k^{(3)} \cdot \{4\left(U_k^{(3)}\right)^2 - \ (-1)^N\} \qquad (3.16)$$

(iii) r = 5

恒等式(3.13)を用いて変形したものは次の通りである：

$$U_{k+1}^{(5)} = U_k^{(5)} \cdot \{16\left(U_k^{(5)}\right)^4 - 20(-1)^N\left(U_k^{(5)}\right)^2 + 5\},$$
$$V_{k+1}^{(5)} = V_k^{(5)} \cdot \{16\left(U_k^{(5)}\right)^4 - 12(-1)^N\left(U_k^{(5)}\right)^2 + 1\}, \qquad (3.17)$$

4．r次収束法の簡単化とその計算効率

前章で行ったようなPell方程式を用いた簡略化を行ったものを一般のrについて示す。以後、混乱の恐れのないときは添え字(r)を省略することにする。

(i) r = even

$$U_{k+1} = \sum_{m=0}^{r/2} a_m \ U_k^{\,r-2m},$$
$$V_{k+1} = U_k \cdot V_k \cdot \sum_{m=0}^{r/2-1} b_m \ U_k^{\,r-2m-2}, \qquad (4.1)$$
$$a_m = (-1)^m \sum_{i=m}^{r/2} \binom{r}{2i}\binom{i}{i-m}, \quad b_m = (-1)^m \sum_{i=m}^{r/2-1} \binom{r}{2i+1}\binom{i}{i-m}$$

(ii) r = odd

$$U_{k+1} = U_k \cdot \sum_{m=0}^{(r-1)/2} a_m \ U_k^{\,r-2m-1},$$
$$V_{k+1} = V_k \cdot \sum_{m=0}^{(r-1)/2} b_m \ U_k^{\,r-2m-1}, \qquad (4.2)$$
$$a_m = (-1)^{Nm+1} \sum_{m=0}^{(r-1)/2} \binom{r}{2i}\binom{i}{i-m}, \quad b_m = (-1)^{Nm+1} \sum_{m=0}^{(r-1)/2} \binom{r}{2i+1}\binom{i}{i-m}$$

以下r次収束法の計算効率を考察する。まずU,Vともにs桁とし、これを初期値としてr次収束法をk回反復したとき、U，Vともにt桁($t = s\,r^k$)の値を得たする。このとき要した計算量を$c_r(s,t)$で表す。反復の1ステップの演算量をU,Vの桁数の二乗に比例するものとして、その比例定数μ_rで表せば$c_r(s,t)$は

$$c_r(s,t) = \mu_r s^2 + \mu_r (rs)^2 + \mu_r (r^2 s)^2 + \cdots + \mu_r (r^{k-1} s)^2$$
$$= \mu_r s^2 \frac{r^{2k}-1}{r^2-1}$$
$$= \mu_r \frac{t^2 - s^2}{r^2 - 1}$$

$$\simeq \frac{\mu_r}{r^2-1}\, t^2 \qquad (4.3)$$

となる。この式の右辺を$c_r(t)$と置く：

$$c_r(t) = M_r t^2, \qquad (4.4)$$

$$M_r \equiv \mu_r/(r^2-1)$$

ここで上式に現れる定数M_r, μ_rを決定するにあたって、いくつかの仮定を設け、またアルゴリズムをより明確にしなければならない。次のような仮定を設ける：

(1) s桁の数とt桁の数の乗算に要する計算量をtsとする。

(2)二乗演算は乗算の半分の手間とする。すなわち、s桁の数の二乗に要する計算量は$s^2/2$である。

(3)多倍長数の乗算、および二乗演算に比べ、多倍長数どうしの加減算、多倍長数×単精度数、等の演算は無視できる。

アルゴリズム(4.1), あるいは(4.2)を計算する場合の手順の概略は以下に示すが、ここではr＝偶数の場合を対象とする。

(1)二乗演算を繰り返すことによって、数列

$$U^2,\ U^4,\ U^8,\ \cdots\cdots,\ U^{2^m},\ m=\lfloor \log_2 r \rfloor \qquad (4.5)$$

を計算する。

(2)これ以外のものは、

$$U^6 = U^2 * U^2$$

$$U^{10} = U^8 * U^2$$

$$U^{12} = U^{10} * U^2$$

$$\vdots$$

として計算する。

(3)Uの必要なベキがすべてそろったところで次のステップのU, Vを求める。

このアルゴリズム1ステップあたりの計算量を考察するにあたって、まずUの必要なベキを求めるのに要する計算量を求める。これは、(i)すべてのベキを乗算によって求めるの要する計算量をA、(ii)式(4.5)のベキを二乗演算によって求めるのに要する計算量をB、(iii)式(4.5)のベキを乗算によって求めるのに要す る計算量をC、とすると

$$A + B - C$$

として求められる。以下、A, B, Cを求める。

本アルゴリズムに現れるのは偶数ベキだけなので、次のような計算を繰り返すことによって、すべてのベキは求まる：

$$U^{2i} = U^{2i-2} \cdot U^2,\quad i=2,3,\ldots,r/2$$

この各々に要する計算量は仮定から$(2i-2)s \cdot 2s$であるから、合計で

$$A \equiv s^2 + \sum_{i=2}^{r/2} (2i-2)s \cdot 2s = (r^2 - 2r + 2)\, s^2/2 \qquad (4.6)$$

だけの計算量を要することになる。一方、式(4.5)のベキをすべて二乗演算を用いて計算すると、

$$B \equiv s^2/2 + (2s)^2/2 + \cdots\cdots + (2^{m-1}s)^2/2$$

$$\hat{=} (r^2-1)\, s^2/6, \qquad (4.7)$$

だけの計算量を要することがわかる。これに対して、これらを乗算で求めるのに要する計算量は

$$C \equiv s \cdot s + 2s \cdot 2s + 6s \cdot 2s + \cdots\cdots + (2^m - 2)\, s \cdot 2s$$

$$= (4r - 4m - 3)\, s^2 \qquad (4.8)$$

```
begin
  {r=even; u[j]=U^j}
   u[1]:=U;
   for  j:=1  to  m  do  u[2^j]:=sqr(u[2^(j-1)]);
   for  j:=2  to  m-1  do begin
          k:=2^j+2;
          repeat
             u[k]:=u[k-2]*u[2];
             k:=k+2
          until  k=2^(j+1)
        und;
   k:=2^m+2;
   repeat
      u[k]:=u[k-2]*u[2];
      k:=k+2
   until  k>r;
   s:=0; t:=0;
   for  j:=0  to  r/2      do  s:=s+a_j・u[r-2j];
   for  j:=0  to  r/2-1  do  t:=t+b_j・u[r-2j-1];
      t:=t*U*V;
      U:=s;
      V:=t
end.
```

図1．r（rは偶数）次収束法のアルゴリズム

である。

最後に、Uの必要なベキから次ステップのVを計算するの要する計算量Dを求める。Vの計算は式

$$V_{k+1}=U_k\cdot V_k\,[b_0U_k^{r-2}+\cdots+b_{r/2-1}]$$

で行われる。上式の［ ］の中は(r－2)s桁と考えられるから、

$$D\equiv s^2+2s\,(r-2)\,s=(2r-3)\,s^2 \tag{4.9}$$

だけの計算量を要する。

以上より、比例定数μ_rは

$$\mu_r=(A+B-C+D)/s^2=(3r^2+r^2+6r-24r+24\log_2 r+5)/6 \tag{4.10}$$

となる。rが奇数のときも同様の解析を行い、

$$\mu_r=(3r^2+r^2-24r+24\log_2 r+20)/6 \tag{4.11}$$

を得る。rの奇偶にかかわらず、

$$\lim_{r\to\infty} c_r(t)=\lim_{r\to\infty}\mu_r t^2/(r^2-1)=2t^2/3 \tag{4.12}$$

という結果を得る。これをM_rについていえば、

$$\lim_{r\to\infty} M_r=2/3 \tag{4.13}$$

である。μ_r，M_rとrの関係を表1に示す。

この表よりr＝5あるいは9あたりが最も効率が良いことがわかる。このことは実験結果にもだいたいの傾向が現れている。なお、ここではU，Vの最終値を求める時間だけを表示し、除算U/Vを行う時間は含まれていない。計算はMELCOM-COSMO 700SのＦＯＲＴＲＡＮで行った。

r	μ_r	Mr	r	μ_r	Mr
2	1.500	0.500	34	675.500	0.585
3	4.500	0.563	35	678.500	0.554
4	9.500	0.633	36	747.500	0.577
5	10.500	0.438	37	750.500	0.549
6	19.500	0.557	38	823.500	0.571
7	22.500	0.469	39	826.500	0.544
8	35.500	0.563	40	903.500	0.565
9	34.500	0.431	41	906.500	0.540
10	51.500	0.520	42	987.500	0.560
11	54.500	0.454	43	990.500	0.536
12	75.500	0.528	44	1075.500	0.556
13	78.500	0.467	45	1078.500	0.533
14	103.500	0.531	46	1167.500	0.552
15	106.500	0.475	47	1170.500	0.530
16	135.500	0.531	48	1263.500	0.549
17	142.500	0.495	49	1266.500	0.528
18	175.500	0.543	50	1363.500	0.546
19	178.500	0.496	51	1366.500	0.526
20	215.500	0.540	52	1467.500	0.543
21	218.500	0.497	53	1470.500	0.524
22	259.500	0.537	54	1575.500	0.540
23	262.500	0.497	55	1578.500	0.522
24	307.500	0.535	56	1687.500	0.538
25	310.500	0.498	57	1690.500	0.520
26	359.500	0.533	58	1803.500	0.536
27	362.500	0.498	59	1806.500	0.519
28	415.500	0.531	60	1923.500	0.534
29	418.500	0.498	61	1926.500	0.518
30	475.500	0.529	62	2047.500	0.533
31	478.500	0.498	63	2050.500	0.517
32	539.500	0.527	64	2175.500	0.531

5．おわりに

本論文では、整数の平方根の連分数近似を加速するアルゴリズムを提案し、その計算効率を考察してきた。本論文で得られた結果をまとめると次のようになる：

（1）本アルゴリズムは除算を含まないため、平方根の有理近似を正確にしかも高速に求めることができる。

（2）本アルゴリズムが生成する有理近似は、基本的には連分数近似であるから常に既約になっている。そのため分母、分子ともに不必要に大きな値にはならない。

（3）計算効率を考察した結果、5次あるいは9次収束法がもっとも効率が良いことがわかった。また収束の次数を上げていくと、計算効率はある値に収束することがわかった。

また、今後の課題としては次のようなことが挙げられる：

（1）分母、分子の最終値を少数展開に変換するとき行う除算の効率をも含めたトータルな計算効率の解析。

（2）乗除算を並列計算で行った場合の計算効率の解析。

（3）乗算をＦＦＴで行った場合の計算効率の解析。

参考文献

1)Hardy,G.H. and Wright,E.M.,"An Introduction to the Theory of Numbers", Oxford Univ. Press, 1979.

2)遠山　啓、"初等整数論"，日本評論社，1972.

3)小沢一文、海野啓明、"$\sqrt{n}$にｒ次収束する有理近似列を生成するアルゴリズムとその誤差解析"，仙台電波高専研究紀要，17(1987)，151-164.

4)和田秀男，"数の世界"，岩波書店，1981.

5)Dutka, J.,"The Square Root of 2 to 1,000,000 Decimals", Math. Comp. 25(1971), 927-930.

6)Traub, J. F.,"Iterative Methods for the Solution of Equations", Chelsa, Pub. Company, 1982.