北海道大学工学部研究報告
第98号（昭和55年）

Bulletin of the Faculty of Engineering,
Hokkaido University, No. 98 (1980)

ニュートン−ラフソン法によるベキ乗算法

柴田俊春* 久郷昌夫*

（昭和54年12月27日受理）

A Decimal Power Function by the Newton-Raphson Method

Toshiharu SHIBATA and Masao KUGO

(Received December 27, 1979)

Abstract

This paper is related to an algorithm for a power function; a well known Newton-Raphson iteration formula used for a n-th root is generalized to power functions of which the powers are decimal numbers.

An introduced algorithm for $x_0^{p_0}$ is constructed as a finite set of equations as,

$$x_{i+1}=(1-p_i)x_{i+1}+\frac{x_i p_i}{x_{i+2}x_{i+1}^{I_{i+1}}} \qquad i=0, 1, \ldots, k$$

$$x_{k+2}=1$$

where constants p and I are decimal and integer numbers, respectively, and can be determined from a next recurrence formula (fractionization),

$$I_{i+1}+p_{i+1}+1=1/p_i, \qquad p_{k+1}=0$$

The required functional value for $x_0^{p_0}$ is given by x_1 which converges to a constant by iterative calculations.

In the above equations, the maximum suffix k depends on figures of p_0 but can be forecasted, and the relation between these can be determined from the probability distributions under the "fractionization". Under the condition that k is large, a somewhat complicated set of equations by the revised fractionization algorithm may reduce the value of "k". Some rates of convergence of x are also illustrated numerically.

1. 緒　　言

計算機によるベキ関数の算出は通常の場合すでに定義されライブラリーとして系に組込まれており，容易にこれを使用出来る状態にある。しかし通常のものは基数が10である十進法での算法に限られている。使用頻度の面からか，従来ベキ関数の指数部が特別の値をとるもの，すなわち平方根，立方根（逆数，平方，立方）は単独に，ほかのベキ関数は対数関数を使用して，算出するのが普通のようである。算法に関しても平方根，立方根，一般的には整数乗根，逆数などは，Newton-Raphson 法を基本にし，これらに加速操作を加えた修正法の研究が多くみられる**。ただし対数関数に関しては Taylor 展開式，最適近似式のほかには基本算法に関する記述が少な

* 応用化学科　第1講座

** 山内，森口，一松："電子計算機のための数値計算法" III（1968）培風館。
細矢治夫：平方根の有理数近似，数学セミナーリーデイングス（1979）日本評論社。

いようである。

もしメモリー使用を極力節約し，かつ精度を任意にあげたいとすれば，対数関数を使用せず，かつ平方根，立方根などの算出を統一した一般化されたベキ関数算法を用意した方がよいと考えられる。

本報告では 10^n 基数（具体的には配列型を念頭においての）にも適用可能な算法の一環としての実数指数のベキ関数算出法を検討したものである。

2. 理　　論

ここで導入しようとするベキ関数は任意の基数と，実数型の指数を念頭におく。算法は内容が理解しやすい Newton-Raphson 法（以下 N-R 法）を用いる。

平方根，立方根，逆数の算出でよく知られているように，まず次の関係によるベキ関数を定義する。

$$x_1 = x_0 ** p_0 \qquad (p_0:\ \text{正の実数})$$

この式の内容はある基数（10 を含めて）に基ずく数 x_0（または配列型数）の実数指数 p_0 乗したものが x_1 であるとするものである。以下簡単のために $p_0<1.0$ とする。もし $p_0 \geq 1.0$ なら p_0 の小数部から出発すればよい。

さて N-R 法によれば

$$f(x_1) = x_1 ** \left(\frac{1}{p_0}\right) - x_0 = 0 \tag{1}$$

となるような x_1 を決めることが出来れば問題が解けたことになる。すなわち近似式として

$$x_1 = x_1 - \frac{f(x_1)}{f'(x_1)} \tag{2}$$

であり，$f'(x_1)$ を具体化して

$$x_1 = (1-p_0)x_1 + \frac{x_0 p_0}{(x_1 ** I_1) * (x_1 ** p_1)} \tag{3}$$

$$I_1 = \mathrm{int}\left(\frac{1}{p_0} - 1\right), \qquad p_1 = \mathrm{mod}\left(\frac{1}{p_0} - 1,\ 1\right) \tag{4, 5}$$

が成立する。(3) 式右辺第 2 項は第 1 項の不正確さを修正する性格のものである。上式中 $p_0 = 1/2,\ 1/3,\ 1/k$ (k: 整数)，-1（特に混乱を生じないので特別に許すことにする）であれば，$p_1 = 0$ であり，それぞれ平方根，立方根，k 乗根，逆数算出の逐次近似式である。この時右辺の x_1 値は旧い値であり，左辺の値は精度が上った新しい値である。p_1 が零であれば，この新しい値 x_1 を逐次右辺に代入し，得られる x_1 が一定値に収斂した時，関数値が決まったことになる。しかし一般的な実数指数の時，p_1 はまだ非零である。したがって x_1 の算出のためには $x_1 ** p_1$ の算出を要する。今この値を $x_2 (= x_1 ** p_1)$ とし，この x_2 も N-R 法で決めることにする。このように N-R 法を重ねて適用するとして，操作を一般化すると次の関係が成立する。

$$x_k = (1-p_{k-1})x_k + \frac{x_{k-1} p_{k-1}}{x_{k+1} x_k ** I_k} \quad ;\ x_{k+1} = x_k ** p_k = x_{k-1} ** (p_{k-1} * p_k) = \ldots \tag{6, 6''}$$

$$x_{k+1} = (1-p_k)x_{k+1} + \frac{x_k p_k}{(x_{k+1} ** p_{k+1}) * (x_{k+1} ** I_{k+1})} \tag{6'}$$

$$I_k + p_k + 1 = 1/p_{k-1}, \qquad I_{k+1} + p_{k+1} + 1 = 1/p_k \qquad (I:\ \text{整数},\ p:\ \text{整数部をもたない小数}) \tag{7, 7'}$$

p_0 から始まり (7) 式で定義される関係により，$p_1, p_2, p_3, \ldots, p_k$ が順次決められる。この p_i を

求める操作を必ずしも適切な用語ではないが“指数の整数化，あるいは分解 (fractionization)”と以下に称する。この指数の整数化操作は容易にわかるように，p_0 が無理数でなければ，有限回の繰返しで零値の p を得る。今 $k+1$ 回目で零になったとすれば $p_{k+1}=0$ であり，(6′) 式は次の形になる。

$$\boldsymbol{x}_{k+1}=(1-p_k)\boldsymbol{x}_{k+1}+\frac{\boldsymbol{x}_k p_k}{\boldsymbol{x}_{k+1}{}^{**}I_{k+1}} \quad ; I_{k+1}+1=1/p_k \tag{8, 8′}$$

以上より $\boldsymbol{x}_1=\boldsymbol{x}_0{}^{**}p_0$ を求めることは次の $(k+1)$ 元連立方程式を解く問題に帰着する。

$$\boldsymbol{x}_{i+1}=(1-p_i)\boldsymbol{x}_{i+1}+\frac{\boldsymbol{x}_i p_i}{\boldsymbol{x}_{i+1}{}^{*}\boldsymbol{x}_{i+1}{}^{**}I_{i+1}} \qquad i=0, 1, \ldots, k \tag{9}$$

ただし $i=k$ のとき $\quad \boldsymbol{x}_{i+2}=\boldsymbol{x}_{k+2}=1.0$

(9) 式はベキ計算としては $\boldsymbol{x}$ の整数乗のみを含み，和差積商の四則演算が保障されているなら，容易に実施可能である。

ところで，任意の実数 p_0 を与えたとき指数の整数化操作，換言すれば何元の連立方程式になるかは，一般に不明である。しかしこの元数は p_0 の桁数に依存するであろうことは容易に想像される。以下これを明らかにするため，整数化とこれに関する性質に数値的な検討を加える。」

3. 指数 p_0 の整数化 (fractionization)

p_0 の実効桁数は簡単なものでは 0.1～0.9 の1桁（これに続く桁は総て零）のものから，2, 3, ..., n, ... 桁と計算機でいえば数語（実数表示）で示される処理可能な限界の数まで拡張することが考えられる。ここでは p_0 の整数化の際の特徴を考え，添字の範囲をどの程度までとればよいかを検討しよう。この問題は整数論の範疇に入る取扱いでもある。今一般に小数以下 n 桁である指数 p_0 をとると

$$10^{-n}\leq p_0=i\cdot 10^{-n}\leq 1-10^{-n} \tag{10}$$

であり，i として1から 10^n-1 まで調べるなら，整数化操作時の特徴を知ることが可能と考えられる。実験は $n=1, 2, 3, 4$ および7について，i の総てにわたって変え，ほかの n のものに対しては $i/10^n$ のいくつかの値をランダムサンプリングで統計的に調べた。調査したものは i, n

---

* ベキ関数処理操作の一般化としては，処理する指数 p_0 が有理数であることから $p_0=m/n$ と分数化し（m, n 整数）

$$\boldsymbol{x}_0{}^{**}p_0=\boldsymbol{x}_0{}^{**}\left(\frac{m}{n}\right)=(\boldsymbol{x}_0{}^{**}m){}^{**}\frac{1}{n} \quad \text{or} \quad =\left(\boldsymbol{x}_0{}^{**}\left(\frac{1}{n}\right)\right){}^{**}m$$

まず $\boldsymbol{x}_0^m$ を求め**，この結果に N-R 法を適用（連立方程式とはならない）するか，あるいは逆に，$\boldsymbol{x}_0$ に $1/n$ の N-R 法を適用し，最終的に m 乗化することも考えられる。手続きとしてはこの方が多元連立方程式を逐次的に解くという操作を必要としないので簡単である。

しかし p_0 の有効数字が多い時，m, n は一般に大きく，大きな数 $(\boldsymbol{x}_0{}^{**}m)$ の n 乗根，あるいは 1.0 に近い $(\boldsymbol{x}_0{}^{**}1/n)$ 値の m 乗というように桁の繰上り下り，が激しく存在する。

計算速度，精度，算法の具体的手続の面からの検討により本報告との優劣を比較する必要があり，後に報告する予定である。

** 指数が整数 (I) であるベキ関数は $\boldsymbol{x}^{**}I$ は I が 2, 8, ... 進で表示されているなら比較的簡単に乗算操作で算出出来る。例えば

$$\boldsymbol{x}^3=\boldsymbol{x}^2{}^{*}\boldsymbol{x}, \quad \boldsymbol{x}^4=\boldsymbol{x}^4, \quad \boldsymbol{x}^5=\boldsymbol{x}^4{}^{*}\boldsymbol{x}, \quad \boldsymbol{x}^6=\boldsymbol{x}^4{}^{*}\boldsymbol{x}^2, \ldots$$

であり右辺の指数はそれぞれ左辺の指数を2進化した時のビットパターンに対応する。すなわち

$$3=(011), \quad 4=(100), \quad 5=(101), \quad 6=(110), \quad \text{etc} \ldots$$

である。したがって $\boldsymbol{x}^{**}I$ の計算には約 $[4.75\log_{10}(1.174\,I)]$ 整数回の乗算を実施すればよい。例えば $\boldsymbol{x}^{100}$ なら判定を含むが約10回の乗算でよい。

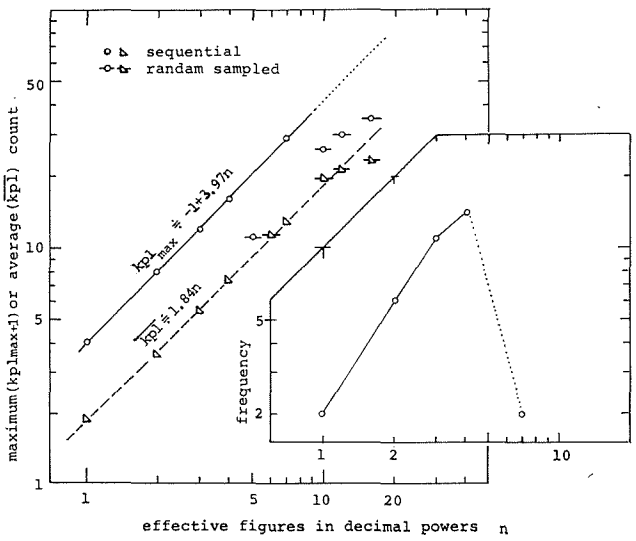

Fig. 1 Maximum and avarage counts at the power-fractionization
An auxiliary figure (right) shows frequency at the maximum count

変更の結果得られる整数化操作の回数（先述 $k+1(=kp1)$）の分布状態，平均回数，最高回数，n を固定し i を変更して行った際の分布の遷移状態などである。

Fig. 1 は横軸に小数点以下の実効桁数 n をとり，その時の p_0 分解繰返回数の最高値 $kp1_{max}$，および平均回数 $\overline{kp1}$ を点綴したものである。図中 *key* で示したように $\overline{kp1}_{max}$ に関しては $n<7$ の全数調査の結果には良好な相関のあることが明らかである。同図には参考のために n を固定し，20～40 個程度の p_0 を用いて得られた $kp1_{max}$ も点綴してある。後者は相関からはずれているが，これは真の $kp1_{max}$ の得られる確率が附図に示したように，n の増加とともに非常に小さくなるので 20～40 個程度の少ない p_0 選定では真の $kp1_{max}$ に遭遇出来なかったことに対応するものと考えられる。p_0～$kp1$ の間には例えば p_0 を連続して変えて行っても，$kp1$ が連続的に変わるのではなく，後に示すように n を指定したとき，$kp1$ 値は確率的な分布相関をとる。なお $n>7$ の $n-kp1_{max}$ 関係などの検討にも p_0 値の全体を調査することが考えられるが，実際には計算時間上の制約から不可能であり，以下に述べるように種々の結果からの推定にたよった（計算時間については処理速度の相違もあろうが，$n=10$ とし p_0 を総て調べるとするなら約 4 年間の計算が必要である）。先に述べた確率分布の存在の可能性は Fig. 1 中に示した $n-\overline{kp1}$ の良好な相関からも推定される。$kp1_{max}$ と異なり $\overline{kp1}$ では任意に選んだ p_0 に対するものでも，相関の延長線上に誤差の範囲内で存在している。

上記のように n の広い範囲にわたり $kp1_{max}$ を相関させることに問題がない訳ではないが，$\overline{kp1}$ の相関性は保障されているものといえよう。さて分布関数の形状に類似性があり，パラメーター間にも相関性があれば分布関数の変量範囲にもやはり相関性が存在し得るものと考えられる。

次に $kp1_{max}$, $\overline{kp1}$ の特徴をより明らかにするために $n-kp1$ 関係を調べた。Fig. 2 は $n=1$～4, 7 に関して総ての p_0 を処理して得られた整数化時の回数 $kp1(=1\sim kp1_{max})$ を離散変量とし，対応する頻度の積算量を求め，これを規格化し，正規確率紙に点綴したものである。図より

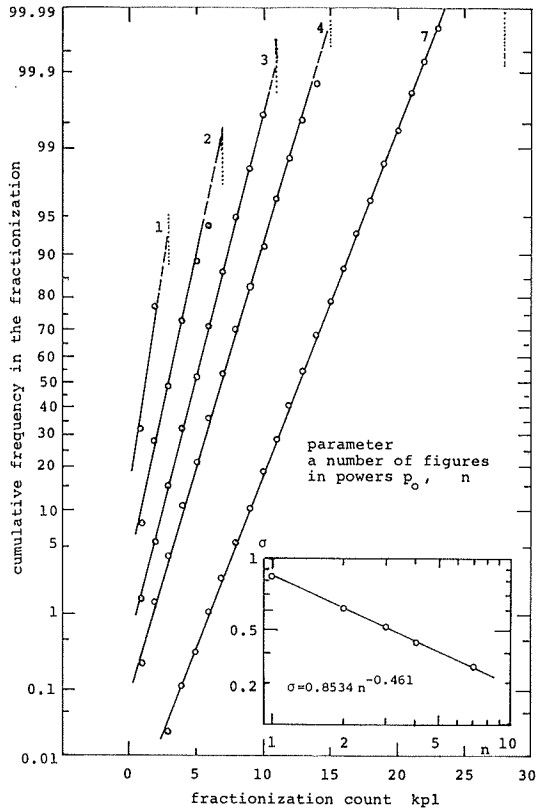

Fig. 2 Effect of effective figures (n) of decimal powers on Gaussian distribution of the fractionization count until no residue and standard deviation vs n

明らかなように $kp1$ の小さい所，および最高の所を除くと良好な直線関係にある*。また図から読みとられる平均値は Fig. 1 のものと少し値は異なるが**，標準偏差と併せて n と良好な相関にあり，分散状態を規定する標準偏差 σ は次式のように整理される。

$$\sigma = 0.8534 \cdot n^{-0.461}$$

n の増加とともに標準偏差が小さくなり $kp1$ の分布が次第に拡がることがわかる。n の増加で $\overline{kp1}$ の増加することは仕方がないとしても，$kp1_{max}$ 推定の立場からは，出来れば分布があまり拡がらないことが望ましい。

さて確率分布と p_0 の関連をさらに検討するため $kp1$ に対し十分に大きな見本点が得られる $n=7$ の場合について，p_0 の全体を 10 個のグループに分割し，それら各々より分布状態の変遷を調べた (Fig. 3)。この結果はグループにより平均値は移動するが，標準偏差はほぼ同一の正規分布を構成し，一連の p_0 中，中心附近以上の指数のグループで大きな $kp1$ を数多く使用されていることがわかった。見本点の総数は異なるが，$n<7$ に関しても同様の傾向がみられた。p_0 の整数化操作とこれに続く逐次近似の連立方程式を解く立場からみると $\overline{kp1}$，$kp1_{max}$ とも当然小さい

* 問題は本報告のものと異なるが，素因数の個数に関する数系列処理（整数論）で正規分布を使用した Erdös-Kac の結果が丸山儀四郎氏によって整数論と確率論の境界問題として紹介されている（数学セミナーリーディング，p. 66 (1979) 日本評論社）。

** 厳密には $kp1$ が小さくなればなるほど Fig. 1 に示した実際の $\overline{kp1}$ と Fig. 2 からの $\overline{kp1}$ との差が大きくなる。したがって n が小さい時には，正規分布というより二項分布に近いことが考えられる。

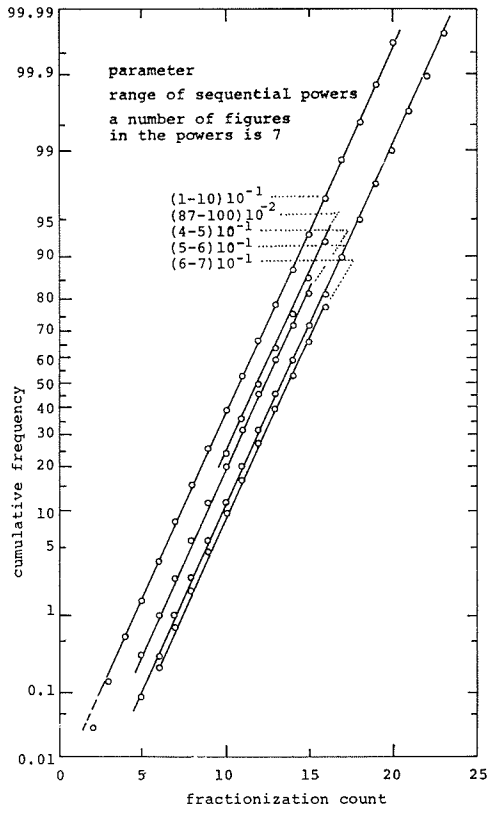

Fig. 3 Transition of the Gaussian distributions for intermediate sequences selected from decimal powers (10^{-7}, 1)

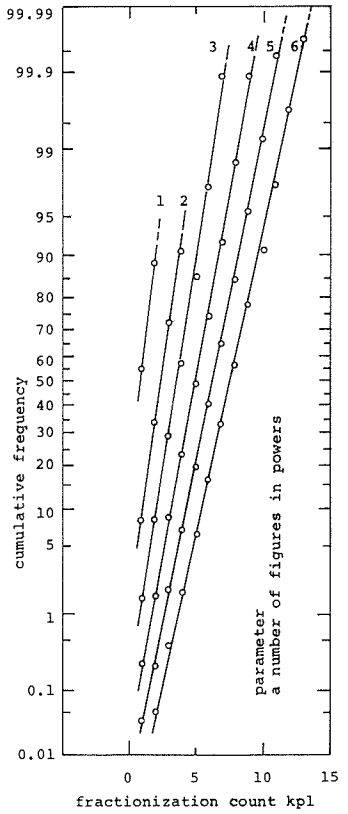

Fig 4 Gaussian distributions of the fractionization count after the revised algorithm

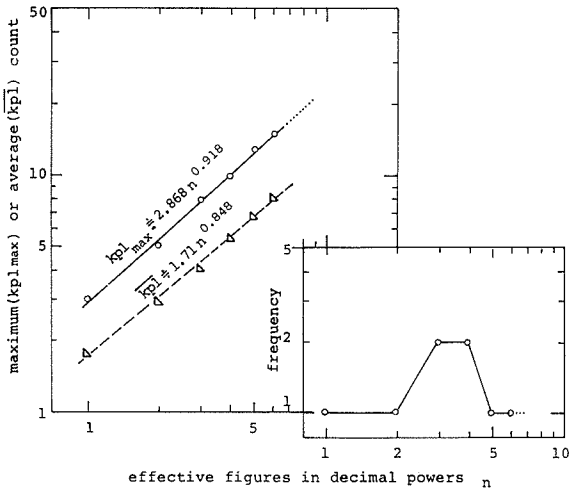

Fig. 5 Maximum and average counts at the power-fractionization — in the case of the revised algorithm — An auxiliary figure (right) shows frequency at the maximum count

ことが望ましい。

ところで $x**p=x/x**(1-p)$ であるから，ベキ関数 $x**p$ を求めることは p の1の補数を指数とする関数値を求めても容易に要求がみたされる。このことは p_0 の分解操作中得られる p_i が0.5より大きければ1の補数をとって新らたに p_i と考えて，分解操作を続ければ，少なくとも $kp1_{max}$ 値だけでも値を減少させることが可能であると考えられる。もちろん $p_i>0.5$ の時，p_i は補数表示に書き直したことの目印をつけておく必要がある。これは p_i を負の値にしておき，以降の p_i 分解は絶対値に関して実施すると約束しておけばよい。x_0**p_0 算出の手順は判断操作を必要とするようになり少し複雑化する。p_i 中補数表示を含む算法を改良法 (rev) と称することにする。もし $kp1_{max}\gg kp1_{max,\,rev}$ または $\overline{kp1}\gg\overline{kp1}_{rev}$ であれば少々の複雑化にかかわらず，計算処理時間の短縮が可能になると考えられる。

さて改良算法による指数 p_0 の分解を上述と類似に実施した結果を Fig. 4 に示す。Fig. 2, 3 で示したと同様，(n が大きいところでは) 明らかに正規分布の形成を示している。さらに先述のように p_0 改良分解法ではいずれも直線の勾配が大きくなり，分布の拡がりが減少し $\overline{kp1}_{max}$ も低下しているのは期待通りである。

Fig. 5 は p_0 改良分解法において得られた $\overline{kp1}$, $kp1_{max}$, ならびに $kp1_{max}$ 時の指数個数である。改良法では $n\geq 7$ 以上の $kp1$ を検討していないが，n を介しての Fig. 1 との対応より，$\overline{kp1}$ は延長線上にあると思われ，$\overline{kp1}_{max}$ もこれに準ずるであろう。

4. p_0 整数化改良法使用時の遂次近似連立方程式の一部修正について

ベキ関数 x_0**p_0 の算出については，第2章ですでに述べたように (7) 式で (I_i, p_i) を求め，連立方程式を解けばよかった。そこでは p_i には整数ではない，非零であるということ以外制限を設けなかった。しかし p_0 分解に改良算法を使用するなら (7), (9) 式は次の形に修正する必要がある。

$$\left.\begin{aligned} 1/|p_{i-1}|-1&=I_i+p_i \quad ;\ p_i>0.5\\ &=(I_i+1)+(p_i-1)\\ &\equiv I'_i+(-p_i)' \quad ;\ p'_i=1-p_i,\quad I'_i=I_i+1 \end{aligned}\right\} \tag{11}$$

$$\begin{aligned} x_i&=(1-p_{i-1})x_i+\frac{x_{i-1}p_{i-1}}{(x_i**I_i)*(x_i**p_i)}\\ &=(1-p_{i-1})x_i+\frac{x_{i-1}p_{i-1}}{(x_i**(1+I_i))*(x_i**(p_i-1))}\\ &=(1-p_{i-1})x_i+\frac{x_{i-1}p_{i-1}x_i**|p'_i|}{x_i**I'_i}=(1-p_{i-1})x_i+\frac{x_{i-1}p_{i-1}x_{i+1}}{x_i**I'_i} \end{aligned} \tag{12}$$

$$x_{i+1}=(1-|p'_i|)x_{i+1}+\frac{x_i|p_i|}{x_{i+2}*x_{i+1}**I_{i+1}} \tag{13}$$

(11)～(13) 式は若干まぎらわしい表現であり，説明を要するかもしれない。まず (11) 式における p_{i-1} の分解に際し，すでに p_{i-2} の地点で $p_{i-1}>0.5$ の結果を経て負号がついているかもしれない。そこで $|p_{i-1}|$ を分解する。この分解の結果 (I_i, p_i) を得たが，$p_i>0.5$ であったとしよう。この時約束により1の補数表示をするので (11) 式右辺第2式のように (I'_i, p'_i) が得られる。$p'_i>0$ に負号をつけ $p_i\equiv -p'_i$, $I_i\equiv I'_i$ と書きかえ (I_i, p_i) を記憶する。以下分解操作を同様に続け $p_{k+1}=0$ まで実施するのは以前述べた通りである。一方逐次近似における (I_i, p_i),

$(p_i<0)$ の使用は定義式 (12) 式右辺第1式を (I_i', p_i') の使用形に直して，第2，第3式を経て第4式のように訂正される。(12) 式中 $\boldsymbol{x}_{i+1}$ は分母にあったものが分子に移っていることに注意すべきである。この $\boldsymbol{x}_{i+1}$ の具体的な値は (13) 式で決められるが，(13) 式の過程では新らしい $\boldsymbol{x}_{i+1}$ が (12) 式中分子で使用されるのか，分母で使用されるのかは無関心である。したがって (13) 式中，唯一の注目点は使用 $p_i'(\equiv p_i)$ が負の値（目印負号）ではなく絶対値をとった正の値で計算しなければならないことである。一般に p_j が負の時のみ $\boldsymbol{x}_j$, $\boldsymbol{x}_{j+1}$ に変更を加え，正であれば先の定義通りの計算サイクルを実施すればよい。

5. 遂次近似連立程式の収斂性

Fig. 5 によれば指数 p_0 が10桁程度の小数なら，最悪で $(kp1_{\max})$ 20元の遂次近似方程式を，平均的には13～4元の方程式を解けばよいことがわかる。なお p_0 が20桁と倍になっても解かれるべき連立方程式の元数は必ずしも多すぎるという程ではない。しかも容易にわかるように，

$$1.0>p_0, p_1, \ldots, p_{j+1}, \ldots$$

$$\boldsymbol{x}_{j+1}=\boldsymbol{x}_0{}^{**}\prod_{i=0}^{j} p_j$$

であるから多元の場合には $\boldsymbol{x}_{j+1}\simeq 1.0$ となり，$\boldsymbol{x}_{j+1}$ の変動は大きくならないのも解を得ることの見通しをよくしていると考えられる。

次に底，指数に値を与え，収斂状態を検討してみる。以下では簡単のために主に p_0 分解を (7) 式で定義する未改良法を中心に述べる。

5.1 指数 p_0 が小さくない場合 $(p_0>0.1)$

簡単な例としてベキ関数 $123^{0.35}$ の計算を実施する。ちなみに $(I, p)_i$ 値は

$$(0,\ 0.35)_0,\quad (1,\ 0.\dot{8}57142\dot{)}_1,\quad (0,\ 0.1\dot{6})_2,\quad (5,\ 0.0)_3$$

であり3元連立方程式である。

この連立方程式の遂次近似解の収斂状態を Fig. 6 に記した。なおここで選んだ $123^{0.35}$ の各数値に，特に深い意味はない。もし改良法を用いるなら，

$$(0,\ 0.35)_0,\quad (2,\ -0.\dot{1}42857\dot{)}_1,\quad (6,\ 0)_2$$

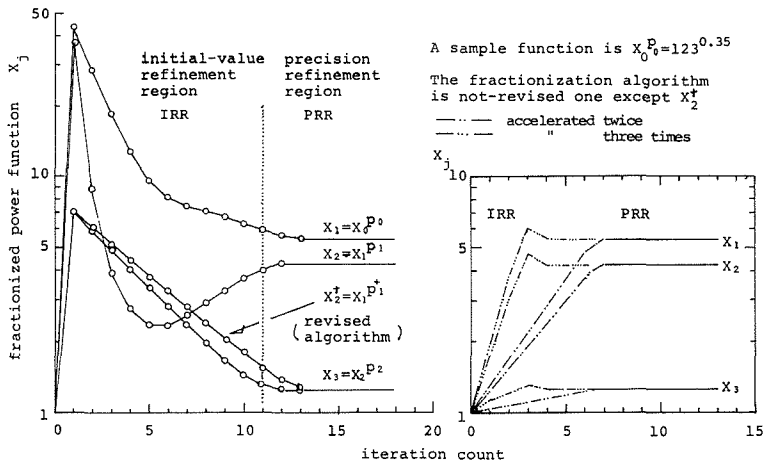

Fig. 6 States of convergence overshoot (left) and hit with some accelerations (right)

であり，2元の連立方程式を構成する(+)。Fig. 6・left において初期値 $\boldsymbol{x}$ は簡単のため1.0と設定し，$\boldsymbol{x}$ の添字の順に計算し，一巡するごとにカウントし，その度に $\boldsymbol{x}$ を取出した変遷を示してある。この初期値設定は必ずしも満足すべき値ではない。したがって最初大きな歪をみせ，近似値 $\boldsymbol{x}$ の値が跳んでいる (overshoot)。しかし計算の繰返しとともに着実に収斂して行くことが示されている。今境界線（点線）で示した繰返し数より小さな領域を初期値調整領域 (IRR)，もう片方の領域を精度高揚領域 (PRR) と称することにする。さて繰返し数を極力小さくするという本来の目的からすれば，不満足な初期値を用いたことによる IRR 領域の減少手段が望まれる。この IRR での overshoot の原因は数値的な検討により，$\boldsymbol{x}_1$ に関する方程式の右辺第2項が過大に寄与したためであることが容易に確かめられる。この overshoot は一般に $\boldsymbol{x}_0$ の増加，p_0 の減少によっても促進される (Fig. 7)。

今 $\boldsymbol{x}_1$ に関し本来もし $I_1 \neq 0$ なら $(1-p_0)\boldsymbol{x}_1 \ll \boldsymbol{x}_0 p_0/\boldsymbol{x}_1{}^{**} I_1 \cdot \boldsymbol{x}_2$ という条件より*, $(1-p_0)\boldsymbol{x}_1$ が

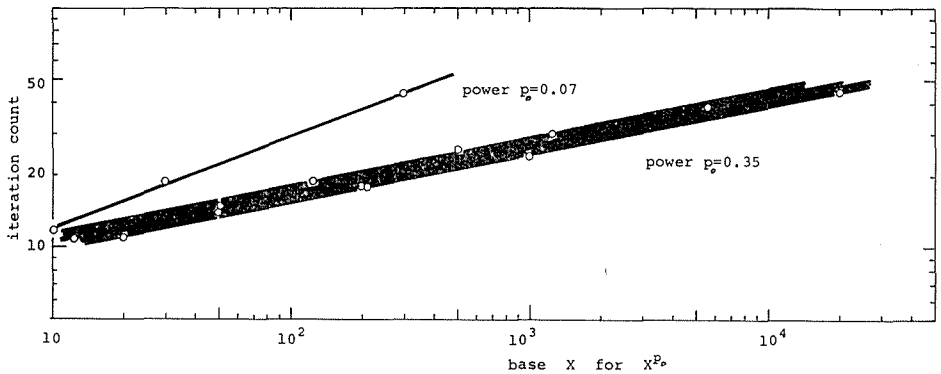

Fig. 7 Increase of iteration counts at the calculation of X^{p_0} because of overshoot
1st approximations were not regulated

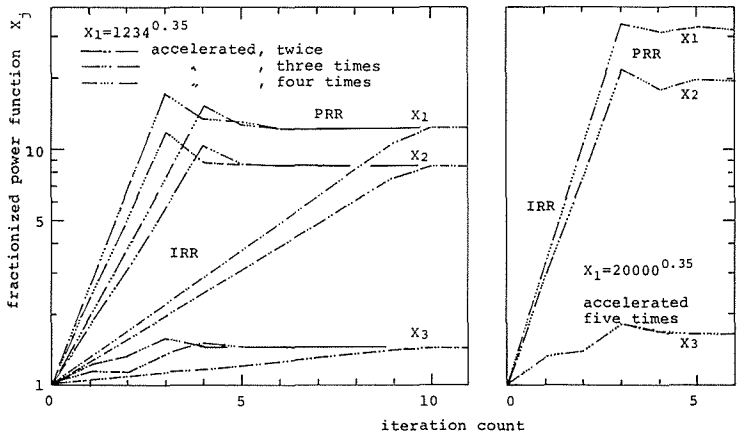

Fig. 8 Changes of iteration counts according to degree of the acceleration
— in a case of an ordinal power —
1st approximates were regulated (not-large overshoot)

---

* $\boldsymbol{x}_1=(1-p_0)\boldsymbol{x}_1+\dfrac{\boldsymbol{x}_0 p_0}{\boldsymbol{x}_1{}^{**}(I_1+p_1)}=\dfrac{(1-p_0)\boldsymbol{x}_1{}^{**}(I_1+p_1+1)+\boldsymbol{x}_0 p_0}{\boldsymbol{x}_1{}^{**}(I_1+p_1)}=\dfrac{(1-p_0)\boldsymbol{x}_1{}^{**}(1/p_0)+\boldsymbol{x}_0 p_0}{\boldsymbol{x}_1{}^{**}(I_1+p_1)}$ $\left(\because\ I_1+p_1+1=\dfrac{1}{p_0}\right)$

$=\dfrac{(1-p_0)\boldsymbol{x}_0+\boldsymbol{x}_0 p_0}{\boldsymbol{x}_1{}^{**}(I_1+p_1)}$ $(\because\ \boldsymbol{x}_1{}^{**}(1/p_0)=\boldsymbol{x}_0)$

if $1-p_0 > p_0$, namely $p_0 < 0.5$ or $I_1 \neq 0$ then $(1-p_0)\boldsymbol{x}_1 > \boldsymbol{x}_0 p_0/\boldsymbol{x}_1{}^{**}(I_1+p_1)$

$x_0 p_0/x_1 ** I_1 * x_2$ より小さいとき x_1 の新らしい近似値として次式を採用する。

$$x_1 = \alpha(1-p_0)x_1 \qquad (\alpha > 1)$$

ここで α は収斂を速める加速係数であり，大きな overshoot を防ぐ役割をもっている。この α を適切に定めることが次の問題である。

α として利用可能で簡単な数は 2.0，あるいは底の有効桁数を利用することが容易に考えられる。このように加速係数を考慮した結果を Fig. 6 の右図に記した。なお加速係数は場合によっては x_1 の繰返し計算中数回にわたって乗ずる必要もある。この結果は IRR 領域での繰返し回数を大幅に逓減し得ることを示している。加速係数の効果については Fig. 8 にも例を示したように底の有効桁数にあわせることで良好な結果が得られるものと考えられる。

5.2 指数 p_0 が小さい場合 ($p_0 < 0.1$)

p_0 が小さい場合にも Fig. 7 で示したように overshoot が存在した。この場合もやはり加速（修正）係数を第1回目の x_1 近似値に乗ずることで効果を発揮する。ただし $p_0 > 0.1$ の場合と異なり，繰返しの適用は必要ない。これは p_0 が小さい時，初期値 1.0 が真の値から遠く離れていないことによる（ただし $x_1, x_2, \ldots$ 間の歪は存在する）。したがって大きな加速使数を適用すると逆に収斂までの繰返し回数が増加する。この様子を Fig. 9 に示した。図中には加速係数 2.0，底の有効桁数，および求めようとする $x_0 ** p_0 (= x_1)$ が有効数字2桁まで正しく指定した場合の例を示した。最良なのは有効数字2桁保障のものであり，次いで加速係数 2.0, ... と収斂性が遅くなる。算出しようとする結果の有効数は一般に不明であるとすれば，加速係数を2ととるのが適当であり，計算に先立って p_0 が小さいか否かを判別する必要があると結論出来る。

以上は主に初期値調整（IRR）について述べてきた。次いでこのように初期値が調整された後の精度高揚領域について述べる。

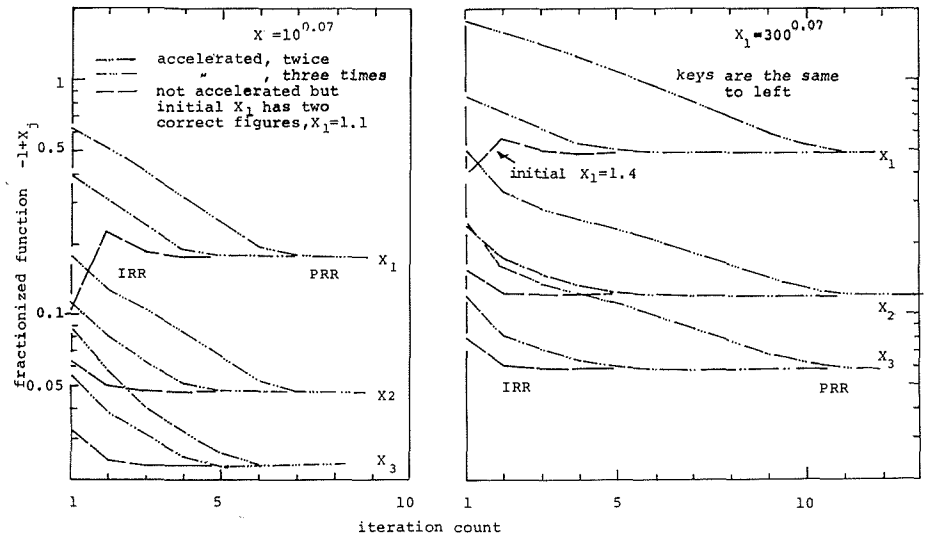

Fig. 9 Changes of iteration counts according to degree of the acceleration
— in a case of a low power and high bases —
(1st approximations were regulated)

5.3 精度高揚領域における収斂について

ベキ計算における最終的な精度と逐次計算の繰返し回数に関しては，従来一般的な報告はみあ

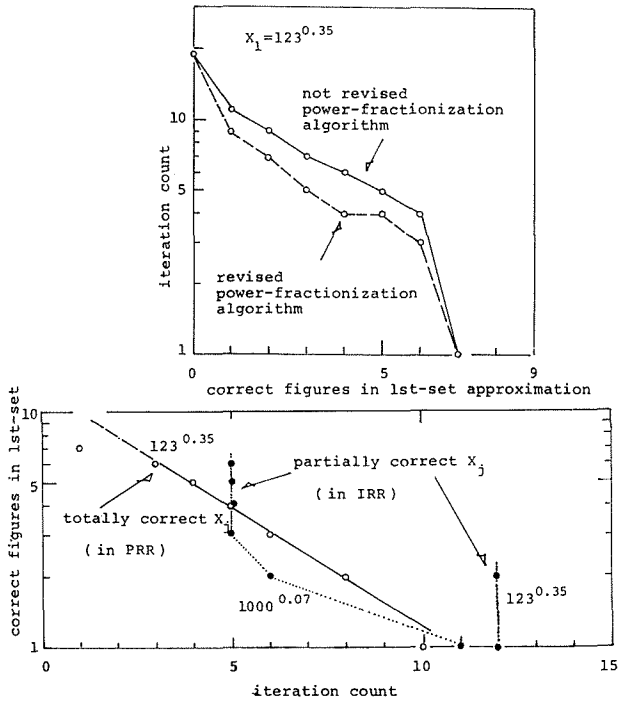

Fig. 10 Effects of correctness in 1st-set for the sequential approximation (final precision of X_6^p is 10 figures)

たらないようである。良く知られた平方根などに関しては収斂性が保障されていること，通常では必ずしもいつもは高い精度を要求しないことの点からか，収斂は非常に速い，あるいは適当な初期値を用いるなら数回の繰返しで収斂値を得ることが可能であるとの記述が普通である。事実筆者らも平方根，および逆数の N-R 法高精度計算で，収斂性の速い（具体的には正しい桁をもつ初期値から始めて（PRR）1回の繰返しで2倍の桁数が正確になる）ことを確認している。ここでは幾桁かが正確である初期値を与え，繰返し数と精度高揚の関係を次の2点について検討した。

1) 初期値の総て $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ にある桁数だけ正しい値を持たせた場合の収斂性

2) 初期値中，最終目的とする $\boldsymbol{x}_1$ のみ（結果的には $\boldsymbol{x}_3$ も正しい桁をもつ）にある桁数正しい値を与え，他の $\boldsymbol{x}$ は1.0として出発した場合の収斂性

最終精度を10桁とした時の実験結果の一例を Fig. 10 に示す。結果より 1) の場合は初期値中の正しい桁の増加とともに繰返し回数の指数関数的減少が示された。なお改良法ではわずかであるが，より収斂性がよい（計算量は元数が少ないことで減るから時間的にみた時，さらに相違が著しくなる)。ただし上述の平方根の場合と比較して1回の繰返し計算で得られる精度の向上割合は，基本指数 p_0 の影響を受けているものと考えられる。2) の場合には指数 p_0 の大小で若干異なった挙動をすることが明らかである。すなわち指数 p_0 が大きい場合，一般的に $p_1, p_2, \ldots$ も小さくなく，$\boldsymbol{x}_1$ に歪が少なくても $\boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$ に含まれる歪のために $\boldsymbol{x}_1$ が最初持っていた精度を打消してしまうことである。一方指数部が小さく底が大きい場合，正しい桁が1個であるか2個であるかにより繰返し回数が極端に異なる。しかしやはり $\boldsymbol{x}_2, \boldsymbol{x}_3$ に含まれる歪のため，$\boldsymbol{x}_1$ のみがいかに正しくても初期値調整領域が必要となり，繰返し回数逓減のためには必ずしもすぐ役立たないことを示している。

6. 結　　論

小数である指数のベキ関数算法を Newton-Raphson 法を用いて表式化した。この際指数の分解法として標準的なもの，および改良したものの２法を提出した。この時いずれも分解に要する操作回数（これは逐次近似の連立方程式の元数でもある）は，指数の桁数によって変わるが，前もって大きさの評価が可能であることを明らかにした。

逐次近似の連立方程式より得られるベキ関数値の収斂は初期値調整領域と精度高揚領域からなり，$p_0>0.1$ では指数の桁数で加速することにより，$p_0<0.1$ では２倍の加速により初期値調整領域の短縮，したがって繰返し回数の逓減が可能であることを示した。

精度高揚のためには近似値間の均衡が必要である。また精度高揚領域で精度は繰返し回数の増加により指数関数的に上昇することを述べた。