巨大数・微小数进行处理するための 算術システムとその2項確率計算への応用

岡山大学教育学部 平 井 安 久

岡山理科大学総合情報学部 中 村 忠

要 旨 1970年代においては、大きな標本数に対する離散型確率分布関数の値の計算は、その表現が簡明であるにもかかわらず、巨大数・微小数の出現と計算時間がかかるという2つの理由により、計算機上での直接計算が困難であった。この欠点を補う方法として、確率分布の近似法と多倍長技法による厳密な計算法が知られている。しかしながら、この2つの方法はそれぞれ固有の欠点を持つ。これらの欠点を避けるため、多倍長技法よりも大きな数値表現が可能な新たな算法をもつ数空間を理論的に構成した。この算法のパーソナルコンピュータ上への実装を目的とした算術アルゴリズムを提案し、それをC言語で実装した。この算術アルゴリズムは数値計算を必要とするあらゆる分野へ応用可能である。本論では、多くの近似法が利用されている2項分布に対して、提案した算術アルゴリズムを適用して、種々の2項確率の計算アルゴリズムを考案した。さらに、それらのアルゴリズムに対して数値実験を行い、精度や計算時間等を考慮して、一番よいものを2項確率の計算アルゴリズムとして提案した。これにより、非常に広範囲な標本数に対して近似法を用いなくても2項確率が計算可能となった。

1. はじめに

選挙調査(例えば、石川(1999))、遺伝学(例えば、中沢(1978))などの分野においては、その分野の特性から、大量なデータが収集される。精密な機械や測定機器が普及し、大量生産の時代である今日において、品質管理や時系列の分野はいうまでもなく、モンテカルロ・シミュレーション(例えば、松本(2002))などの分野においても大量なデータが収集される。最近の通信技術(インターネット、人工衛星等)の発展に伴い、多くの分野で大量データが即座に収集可能になっている。特に、データマイニングあるいはWebmining(例えば、Gaul(2005))と呼ばれる解析手法が近年注目されてきたことからわかるように、インターネットを通じて収集される大規模データの統計解析法が研究され、実用化され始めている。近年、個人がパーソナルコンピュータ(以下パソコンという)を持てるといっていいほど普及しているし、ほとんどのパソコンユーザは世界規模で広まっているインターネットを利用している。従って、個人でも大規模データの収集が容易にできるので、大量データを統計解析する機会が増している。このような状況のもとで、大量データに対する確率値を計算することは日常的なこととなりつつある。ここでは離散型データのみを扱

うことにする。

一方、コンピュータによる離散型確率分布の確率関数や分布関数の計算については、1940年代から行われ始めた(詳細は第3節参照)。離散型確率分布の確率関数や分布関数の値を、近似式ではなく、その数式表現に従ってコンピュータで計算する方法を直接確率計算法ということにする。当時のコンピュータは数値演算処理速度が遅く、また個人がコンピュータを持てる状況ではなかった。そのため、直接確率計算法の代替的方法として理論から導出される近似式と“理論的に導出される近似分布(例えば、正規分布)の確率分布表”の組を用いて計算する方法(近似確率計算法という)が採用されるようになった。近年、パソコンでも数値演算処理はかなり高速になってきたが、コンピュータの構造的欠点であるオーバーフロー・アンダーフロー問題は解決されていない。そのため、何らかの工夫をしない限り、多くの離散型確率分布の確率関数の値のコンピュータによる直接確率計算は一般的には困難である。一方、近似確率計算法は、オーバーフロー・アンダーフローの発生を部分的に回避するアルゴリズムとなっているので、研究者から一般の統計利用者まで広く使用されている。また、大きな整数の四則演算をするための多倍長計算技法(例えば、Knuth (1981))もオーバーフロー・アンダーフローの発生を部分的に回避するアルゴリズムとなっている。この理由のため、この技法を実装したフリーまたは有償の計算ソフトウェアは多数ある。これらの中で、数式処理ソフトウェア Mathematica, Maple が小さい数からかなり大きい数まで広範囲な数で計算可能な算術システムを実現している。しかしながら、数式処理ソフトウェアは高価なので、経済的理由により入手困難な場合がある。また、多倍長計算技法は高精度な計算のため、コンピュータのメモリを多量に消費したり、計算時間がかかる等の欠点がある。

これを動機として、少数データから巨大データまで広い範囲でパソコンによる直接確率計算が可能な算術理論の構築とその算術アルゴリズムの開発を行うこと、およびそれらが無償で公開することが本稿の目的である。下記のホームページには、今回開発した2項確率計算プログラムが実行可能なファイルとして公開され、第3節での数値実験の詳細な結果も公開されている(http://ed-www.ed.okayama-u.ac.jp/~suugaku/newhira/stat2006/mogura_webStart.htm)。

以下、第2節では、上記目標を達成するために、新しい算術を理論的に構成し、その算術システムを構築するための算術アルゴリズムを提案する。この新しい算術はすべての数値計算に適用可能である。第2.1節ではパソコン上で広範囲の実数を表現するため、実数全体集合と代数的に同型な集合(新しい数体系)を構成し、この集合上に算術(四則演算)を導入する。この算術をモグラ算術といい、この集合の元をモグラ数という。モグラ数は通常の実数に代わる新しい数の表現となる。この新しい数表現を用いると、コンピュータ上で広範囲の実数を表現できる。従って、巨大数あるいは微小数を処理できるということになる。第2.2節では、前節で導入されたモグラ算術をパソコンで作動する処理系で実現できるモグラ算術アルゴリズムを導入する。この算術アルゴリズムは浮動小数点算術演算の長所はそのまま利用し、欠点を補うように構成されている。処理系としてC言語を採用するならば、理論上、扱える数の絶対値の最大値は約10の $1.378101\text{E}18$ 乗、扱える数の絶対値の最小値は約10の $-1.378101\text{E}18$ 乗である。モグラ算術アルゴリズムはあらゆる数値計算に応用可能である。後節で必要となるいくつかの関数のモグラ算術アルゴリズムを導入する。

第3節では、第2節で導入されたアルゴリズムを2項確率関数および2項分布関数の計算に応用する。広い範囲の標本の大きさに対して2項分布の確率関数と確率分布関数の値を計算できる直接確率計算アルゴリズムを提案する。第3.1節では2項確率関数の直接確率計算アルゴリズム

を4種類導入する。それぞれの手法に対して数値実験を行い、絶対誤差、相対誤差などの精度保証および計算時間等を基準に一番よい方法を2項確率関数の値の計算アルゴリズムとして提案する。第3.2節では、2項分布の確率分布関数の直接確率計算アルゴリズムとして2通りの方法が提案される。これらのアルゴリズムにはある種の高速演算が組み込まれている。第3.3節では、第3.2節で提案された2つの方法の数値実験を行う。絶対誤差、相対誤差などの精度保証および計算時間等を基準にして比較・検討する。その結果、良いと判断される方法を確率分布関数の値の計算アルゴリズムとして提案する。本論での数値実験は Borland C++ (ver.5.0J) と Pentium 4 (2.66 GHz) が装着されたパソコンを使用して行った。最後に、各種の数値実験で得られた結果や今後の課題をまとめて結びに代える。

2. モグラたたき算術とそのアルゴリズム

2.1. 新しい数体系

パソコンにおける数の表現や算術演算は2進表示を用いた不動小数点演算を基礎にして構成されている。この表示方法は32ビット(単精度)または64ビットフォーマット(倍精度)で実数を表現しているため、大きな数(または小さな数)は表示できないあるいは計算不可能というオーバーフロー(またはアンダーフロー)問題が生じる。パソコンに標準で装備されている算術システムの長所はそのまま利用し、短所を補う新しい算術アルゴリズムを構築するため、実数体系と代数的に全く同じ数の体系を導入する。この体系の元を実数の別表現として扱う、すなわち、この数体系に算術が導入される(定理1参照)。以下この数体系とその算術をどのように導入するかを述べよう。集合 $\mathcal{N}$ は整数全体集合とし、正整数 M を任意に固定する。次の集合

$$\mathcal{M} = ([-10^M, -10^{-M}] \cup \{0\} \cup [10^{-M}, 10^M]) \times \mathcal{N} \quad (1)$$

を考える。集合 $\mathcal{M}$ に関係 ($\equiv$) を導入する。

定義1 $(s, i), (t, j) \in \mathcal{M}$ とする。 $s \times 10^{M \times i} = t \times 10^{M \times j}$ のとき、 (s, i) と (t, j) の間には関係 ($\equiv$) があるといい、記号で、 $(s, i) \equiv (t, j)$ とかく。

この関係 ($\equiv$) は同値関係である。元 (s, i) と同値な関係にある元の全体 $\{(t, j) \in \mathcal{M}; (s, i) \equiv (t, j)\}$ (同値類という) を記号 $\langle s, i \rangle$ で表す。例えば、同値類 $\langle 1, 0 \rangle$ は集合 $\{(1, 0), (10^{-M}, 1), (10^M, -1)\}$ を、同値類 $\langle 2, 2 \rangle$ は集合 $\{(2, 2), (2 \times 10^{-M}, 3)\}$ を表す。

集合 $\mathcal{M}$ を同値関係 ($\equiv$) による同値類に類別し、その同値類の全体を $\mathcal{M}/\equiv$ とかく。集合 $\mathcal{M}/\equiv$ に2種類の算法(加法, 乗法)を導入するために次の変換を準備する。

$$I(z) = \begin{cases} 0, & z = 0, \\ -\min\{k; 10^{-M} \leq |z| \times 10^{Mk} \leq 10^M, k = 0, 1, 2, \dots\}, & 0 < |z| \leq 1, \\ \min\{k; 10^{-M} \leq |z| \times 10^{-Mk} \leq 10^M, k = 0, 1, 2, \dots\}, & |z| > 1, \end{cases} \quad (2)$$

$$A(z) = \begin{cases} 0, & z = 0, \\ z \times 10^{-M \times I(z)}, & z \neq 0. \end{cases} \quad (3)$$

定義より、任意の実数 z に対して一意に $(A(z), I(z))$ が定まり、 $(A(z), I(z)) \in \mathcal{M}$ かつ $z = A(z) \times 10^{M \times I(z)}$

と書けることがわかる．この変換を用いて加法 ($\oplus$), 乗法 ($\otimes$) を次のように定義する．

定義 2 $\langle s, i \rangle, \langle t, j \rangle \in \mathcal{M}/\equiv$ とし, $w = s \times 10^{Mi}, z = t \times 10^{Mj}$ とする．

(i) 加法 ($\oplus$) : $\langle s, i \rangle \oplus \langle t, j \rangle = \langle A(w+z), I(w+z) \rangle$.

(ii) 乗法 ($\otimes$) : $\langle s, i \rangle \otimes \langle t, j \rangle = \langle A(w \times z), I(w \times z) \rangle$.

このとき, 2つの算術 $\oplus, \otimes$ を持つ集合 $\mathcal{M}/\equiv$ は次のような代数的性質を持つ (詳細は平井・中村 (2004a) を参照) .

定理 1 集合 $\mathcal{M}/\equiv$ は実数体 $\mathcal{R}$ と同型な可換体である．

上の定理により, $\langle s, i \rangle (\in \mathcal{M}/\equiv)$ を実数の1つの表現とみなせる．加法 ($\oplus$) に関する単位元は $\langle 0, 0 \rangle$ であり, 乗法 ($\otimes$) に関する単位元は $\langle 1, 0 \rangle$ である． $\langle s, i \rangle, \langle t, j \rangle \in \mathcal{M}/\equiv$ とする．加法 ($\oplus$) に関する $\langle t, j \rangle$ の逆元を $-\langle t, j \rangle$ とする． $\langle s, i \rangle \oplus (-\langle t, j \rangle)$ を $\langle s, i \rangle \ominus \langle t, j \rangle$ で表し, これを $\mathcal{M}/\equiv$ における減法という．また, $\langle 0, 0 \rangle$ と異なる $\langle t, j \rangle$ の乗法 ($\otimes$) に関する逆元を $\langle t, j \rangle^{-1}$ とする． $\langle s, i \rangle \otimes \langle t, j \rangle^{-1}$ を $\langle s, i \rangle \oslash \langle t, j \rangle$ で表し, これを $\mathcal{M}/\equiv$ における除法という． $-\langle t, j \rangle = \langle -t, j \rangle, \langle t, j \rangle^{-1} = \langle 1/t, -j \rangle$ だから, 減法 ($\ominus$), 除法 ($\oslash$) は次のようにもかける．

(i) 減法 ($\ominus$) : $\langle s, i \rangle \ominus \langle t, j \rangle = \langle s, i \rangle \oplus \langle -t, j \rangle$.

(ii) 除法 ($\oslash$) : $\langle s, i \rangle \oslash \langle t, j \rangle = \langle s, i \rangle \otimes \langle 1/t, -j \rangle$.

一方, $w = s \times 10^{Mi}, z = t \times 10^{Mj}$ とおけば, 明らかに $\langle s, i \rangle \ominus \langle t, j \rangle = \langle A(w-z), I(w-z) \rangle$, $\langle s, i \rangle \oslash \langle t, j \rangle = \langle A(w/z), I(w/z) \rangle$ が成立する．

$\mathcal{M}/\equiv$ に導入された算術は2つの特徴をもっている．第一に, 通常の実数の演算結果として得られる数の絶対値がある正数 (10^M) を越えたら, それより小さくなるように調整する．これはオーバーフロー問題が起こりにくい操作になっている．逆に通常の実数の演算結果として得られる数の絶対値がある正数 (10^{-M}) より小さくなったら, それより大きくなるように調整する．これはアンダーフロー問題を起こしにくい操作になっている．第二に, 実数である範囲内の実数と整数の組で表現するため, コンピュータでの数表現の範囲が通常と比べて格段に広いということである (第2.2節参照) . 上記のような特徴から, この算術を“モグラたたき算術”, $\mathcal{M}/\equiv$ の元 $\langle s, i \rangle$ をモグラ数と呼ぶことにする．

2.2. モグラ技法

2.1節で考察した数体系 $\mathcal{M}/\equiv$ のモグラたたき算術は算術論理演算 (ALU) が装置されているコンピュータで作動し, かつ倍精度実数型の数を処理できる任意の処理系 (C, FORTRAN, PASCAL, JAVA 等) で実現可能である．従って, コンピュータで作動する数式処理システム (Mathematica, Maple 等) の上でも実現可能である．この算術をC言語に限って実現するが, 他の処理系でも同様にしてできる．本節ではC言語をもちいてどのようにモグラ数 $\langle s, i \rangle (\in \mathcal{M}/\equiv)$ およびモグラ算術を実現するかを述べよう．本論で提案する算術システムはオーバーフロー・アンダーフローが起こりにくいように巨大数および微小数を処理することが目的である．以下, $M = 153$ として議論する． $10^{-M} \leq |s| \leq 10^M$ の範囲の実数 s を倍精度実数型で表したものを $\tilde{s}$ とする．C言語の倍精度表現で正確に表現可能な整数の最大値は $D = 9007199254740991$ である． $-D \leq i \leq D$ の範囲の整数 i を倍精度実数型で表したものを $\tilde{i}$ とする．実数 s および整数 i が, $10^{-M} \leq |s| \leq 10^M$

および $-D \leq i \leq D$ の範囲にあるとき、モグラ数 $\langle s, i \rangle$ の C 言語による実現として、 $\langle \tilde{s}, \tilde{i} \rangle$ を採用する。このような、 $\langle \tilde{s}, \tilde{i} \rangle$ の全体に $\langle 0, 0 \rangle$ を加えた集合 C を $M/\equiv$ の実現として採用する。本部分節以降では、混乱の恐れがない限り、 $\langle \tilde{s}, \tilde{i} \rangle$ を単に $\langle s, i \rangle$ で表し、 $M/\equiv$ の C 言語による実現 C は次のように表す。

$$C = \{\langle s, i \rangle; 10^{-M} \leq |s| \leq 10^M, -D \leq i \leq D\} \cup \{\langle 0, 0 \rangle\}.$$

C 言語に標準で装備されている算術システムでは、非常に大きな(または小さな)数 $s \times 10^{Mi}$ を表現(あるいは処理)できないが、モグラ表現では数 $\langle s, i \rangle (\in C)$ と表現できる。

C 言語を使用した四則演算アルゴリズムを導入するために若干の準備をする。ここで導入する算術アルゴリズムは入力される倍精度型実数 r を直接処理するのではなく、実数 $A(r)$ と整数 $I(r)$ の組 $\langle A(r), I(r) \rangle (\in C)$ に変換し、この組に対して C に導入される四則演算を実行するのである。前節で述べたようにモグラ数 $\langle A(r), I(r) \rangle$ と(倍精度型)実数 r は理論的には同じ実数である。しかし、両者は異なる算術システムにおいて計算が実行される。すなわち、倍精度型実数 r は C 言語に標準で装備されている算術システムで、モグラ数 $\langle A(r), I(r) \rangle$ は本節で提案されるモグラたたき算術システムで計算が可能となる。

C に次のような四則演算を導入する。混乱の恐れがない限り、演算の記号は $M/\equiv$ における演算 $(\oplus, \ominus, \otimes, \oslash)$ と同じ記号を用いる。一方、記号 $(+, -, \times, /)$ は処理系(ここでは C 言語)に標準で装備されている算術システムにおける四則演算記号を表す。 $\langle s, i \rangle, \langle t, j \rangle \in C$ とする。

乗法 ($\otimes$): $\langle s, i \rangle \otimes \langle t, j \rangle = \langle A(s \times t), i + j + I(s \times t) \rangle.$

除法 ($\oslash$): $t \neq 0$ とする。

$$\langle s, i \rangle \oslash \langle t, j \rangle = \langle A(s/t), i - j + I(s/t) \rangle.$$

加法 ($\oplus$):

(a) $s = 0$ または $t = 0$ の場合:

$$\langle s, i \rangle \oplus \langle t, j \rangle = \begin{cases} \langle s, i \rangle, & t = 0, \\ \langle t, j \rangle, & s = 0. \end{cases}$$

(b) $s \neq 0, t \neq 0$ の場合:

(b1) $i = j$ の場合:

$$\langle s, i \rangle \oplus \langle t, j \rangle = \begin{cases} \langle s + t, i \rangle, & st < 0, \\ \langle A(s + t), I(s + t) + i \rangle, & st > 0. \end{cases}$$

(b2) $i \neq j$ の場合:

(b21) $i - j = -1$ の場合:

$$\langle s, i \rangle \oplus \langle t, j \rangle = \langle A(s \times 10^{-M} + t), I(s \times 10^{-M} + t) + j \rangle.$$

(b22) $i - j = 1$ の場合:

$$\langle s, i \rangle \oplus \langle t, j \rangle = \langle A(s + t \times 10^{-M}), I(s + t \times 10^{-M}) + i \rangle.$$

$$(b23) |i - j| \neq 1 \text{ の場合: } d_s = i + M^{-1} \log_{10} |s|, d_t = j + M^{-1} \log_{10} |t|, \\ \kappa = 1 - 10^{-16} \text{ とおく.}$$

$$\langle s, i \rangle \oplus \langle t, j \rangle = \begin{cases} \langle s, i \rangle, & d_t - d_s \leq -\kappa, \\ \langle t, j \rangle, & d_t - d_s \geq \kappa, \\ \langle s, i \rangle \otimes \langle 1 + (t/s) \times 10^{M(j-i)}, 0 \rangle, & d_t < d_s < d_t + \kappa, \\ \langle t, j \rangle \otimes \langle 1 + (s/t) \times 10^{M(i-j)}, 0 \rangle, & d_s < d_t < d_s + \kappa. \end{cases}$$

減法 ($\ominus$): $\langle s, i \rangle \ominus \langle t, j \rangle = \langle s, i \rangle \oplus \langle -t, j \rangle$.

以下, この算術アルゴリズムによって各種アルゴリズムやプログラムを作る方法をモグラ技法とよぶ.

(注 1) M の値として 153 を採用している. 精度や計算時間という観点からみたら, $M = 153$ とすることは本質的ではない. 本論の 3.1 及び 3.3 節で行う数値実験においては, $M = 7$ でも $M = 153$ の場合とほぼ同程度の精度や計算時間である (平井・中村 (2004a)). $M = 7$ に対応する算術システムと $M = 153$ に対応するそれが本質的に違う点は扱える数の範囲が後者の方が広いということである. 例えば, 扱える最大の桁数は前者が 63049394783186945 桁に対し, 後者は 1378101485775371776 桁である. $M = 153$ を採用した理由は, 現在普及している C/C++ コンパイラが 32 ビット対応であるためである. 64 ビット対応のプロセッサが市場に出始めたのをきっかけに, アプリケーションソフト (特に C/C++ コンパイラ) の 64 ビット化が始まっている. 本論で提案している算術システムは M の値を変えることで 64 ビット C/C++ コンパイラにも対応可能である.

モグラ技法は四則演算が定義されているのであらゆる数値計算に応用可能である. また, 上記の各演算において, 得られるモグラ数 $\langle s, i \rangle$ の第 2 成分の整数 i が $[-D, D]$ の範囲を超えているときは, この技法でもオーバーフローあるいはアンダーフローが起こることになる. モグラ数 $\langle s, i \rangle$ は $10^{-M} \leq |s| \leq 10^M, -D \leq i \leq D$ を満たすから, 10 進数でいうと符号を除いて $10^{-M(D+1)}$ から $10^{M(D+1)}$ までの表現が可能である. 従って, この技法の利点は通常の C 言語に比べてオーバーフローあるいはアンダーフローが非常に起こりにくくなることである. この技法の計算精度は C 言語の持っている精度と同じなので, 巨大な回数 (例えば数十億回) の繰り返し演算は, 通常の C 言語の算術システムと同じく, 精度の低下を起こす可能性がある. また, モグラ数 $\langle s, i \rangle$ において, 第 2 成分の整数 i が巨大になると, モグラ数から通常の C 言語の算術システムで扱える倍精度型実数には変換できない. 逆はいつでも可能である. よく知られた数学関数は上で述べた算術アルゴリズムで作成可能である. ここでは, 第 4 節以降の議論に使用する関数の作成例をいくつか述べよう (詳細は平井・中村 (2004a) 参照). $\langle s, i \rangle \in C$ とする.

(i) 底が 10 のモグラ対数関数 $\mathbf{mlog10}(\langle s, i \rangle)$

$$\mathbf{mlog10}(\langle s, i \rangle) = \langle A(\log_{10} s + Mi), I(\log_{10} s + Mi) \rangle, s > 0.$$

(ii) モグラ数の整数べき乗 $\mathbf{mpowi}(\langle s, i \rangle, n)$

最初に整数 n が正で, 実数 s が正である場合を考える. 10 進数の正整数 n の 2 進数表示

を $n = (\alpha_\ell \alpha_{\ell-1} \cdots \alpha_1 \alpha_0)_2$ とする. 常に $\alpha_\ell = 1$ であることに注意しよう. $\hat{s} = s \times 10^{-[\log_{10} s]}$ とおく. 任意の正数 x に対して, $1 \leq x \times 10^{-[\log_{10} x]} < 10$ であるから, $1 \leq \hat{s} < 10$ かつ $\langle s, i \rangle = \langle \hat{s}, 0 \rangle \otimes \langle 10^{[\log_{10} s]}, i \rangle$. ただし, $[z]$ は z を超えない最大の整数を表す. $\mathbf{z}_0 = \langle \hat{s}, 0 \rangle$ とおく. $j(= 1, 2, \dots, \ell)$ に対して

$$\mathbf{z}_j = \langle \hat{s}^{\alpha_{\ell-j}}, 0 \rangle \otimes \mathbf{z}_{j-1} \otimes \mathbf{z}_{j-1}$$

を繰り返す. $\mathbf{z}_\ell$ が $\langle \hat{s}, 0 \rangle$ の n 乗 $\langle \hat{s}, 0 \rangle^n$ に対応するモグラ表現である. これは実数のべき乗の高速計算法としてよく知られている方法(例えば, Knuth (1981))のモグラ版である. $n \times [\log_{10} s]$ を M で割ったときの商を $Q(\geq 0)$, 余りを $R(0 \leq R < M)$ とする. これと上記の繰り返し計算により得られた $\mathbf{z}_\ell$ を用いて,

$$\mathbf{mpow1}(\langle s, i \rangle, n) = \mathbf{z}_\ell \otimes \langle 10^R, ni + Q \rangle$$

とおく. $\mathbf{z}_j = \langle r_j, k_j \rangle$ ($j = 1, 2, \dots, \ell$) とすれば,

$$\mathbf{mpow1}(\langle s, i \rangle, n) = \langle A(r_\ell 10^R), k_\ell + I(r_\ell 10^R) + ni + Q \rangle$$

となる. モグラ表現 $\mathbf{z}_j = \langle r_j, k_j \rangle$ ($j = 1, 2, \dots, \ell$) の成分については, 前節で述べた $I(r)$, $A(r)$ の性質により, $1 \leq r_j \leq 10^M, k_j \geq 0$ ($j = 1, 2, \dots, \ell$) となることに注意しよう. 以上の準備の下で, 一般の整数 n と, 実数 s に対して, $\langle s, i \rangle$ の整数べき乗 $\langle s, i \rangle^n$ の値を算出する関数を次式で定義する.

$$\mathbf{mpowi}(\langle s, i \rangle, n) = \begin{cases} \mathbf{mpow1}(\langle s, i \rangle, n), & n > 0, s > 0, \\ \langle (-1)^n, 0 \rangle \otimes \mathbf{mpow1}(\langle -s, i \rangle, n), & n > 0, s < 0, \\ \langle 1, 0 \rangle, & n = 0, s \neq 0, \\ \langle 0, 0 \rangle, & n \neq 0, s = 0, \\ \langle 1, 0 \rangle \otimes \mathbf{mpow1}(\langle s, i \rangle, -n), & n < 0, s > 0, \\ \langle (-1)^n, 0 \rangle \otimes \mathbf{mpow1}(\langle -s, i \rangle, -n), & n < 0, s < 0. \end{cases}$$

同様に, モグラ数の実数べき乗 $\mathbf{mpowr}(\langle s, i \rangle, t)$, モグラ数のモグラべき乗 $\mathbf{mpow}(\langle s, i \rangle, \langle t, j \rangle)$ を定義することができる. 一方, 階乗 $n!$ については, 3 種類の関数 $\mathbf{mfact0}(n)$, $\mathbf{mfact1}(n)$, $\mathbf{mfact2}(n)$ を導入する. $\mathbf{mfact0}(n)$ は $1 \times 2 \times \cdots \times n$ を直接計算したものである. $\mathbf{mfact1}(n)$ は, 100 万までは 10 万ごとに, 5 億までは 50 万ごとにあらかじめ(モグラ技法四則計算により)計算された階乗値(1008 個の数値からなるデータセット)を用いて定義される. これにより, n が 5 億まで $\mathbf{mfact1}(n)$ の計算が可能になる. n が 5 億より大きいときは, データセットが利用できないので, $\log \Gamma(z)$ の近似式(例えば, Abramowitz and Stegun (1972), 森口・宇田川・一松 (1995))を用いる. まず, $\log \Gamma(z)$ に対応する関数 $\mathbf{mloggam}(\langle s, i \rangle)$ を導入し, これを利用して $\mathbf{mfact2}(n)$ を $\mathbf{mfact2}(n) = \mathbf{mpow}(\langle e, 0 \rangle, \mathbf{mloggam}(\langle s, i \rangle))$ と定義することができる.

3. 2 項分布への応用

離散型確率分布は無数にあるため, 本論では最も古くから知られ, かつ数多くの近似確率計算法が開発されている 2 項分布に焦点をあてて, 2 項確率関数の値と 2 項確率分布関数の値(以下, 2 項確率

という)を算出するアルゴリズムを提案する. 2項分布 $B(n, p)$ の確率関数 $b(x; n, p)(= nCx p^x q^{n-x})$ および2項確率

$$B(x; n, p) = \sum_{0 \leq i \leq x} b(i; n, p), \quad x = 0, \dots, n \quad (4)$$

は実数の四則演算を有限回行って計算される単純な量である. その形の単純さにもかかわらず, n が大きいとき, これらの値は, 直接確率計算法ではなく, 近似確率計算法を用いて計算される. 例えば, 2項確率関数の場合, Prohorov の近似式 (Johnson et al. (1992))

$$b(x; n, p) \approx \frac{1}{\sqrt{npq}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} \frac{(x - np)^2}{npq}\right)$$

がよく利用される.

一方, ある程度大きな n に対して2項確率を求める計算アルゴリズムは1940年代から論じられてきた. 1950年代からコンピュータによる2項確率の直接確率計算が行われ, 2項確率表として公表されるようになった(例えば, Robertson (1960), Weintraub (1963), Stuart and Ord (1994, p.166)). これらの2項確率表は, 高々1000くらいまでの標本の大きさ n と $1/1000$ 程度までの小さな p のいくつかの組 (n, p) に対してのみ計算されたものであり, 一般的な組 (n, p) に対する2項確率表としての実用性には欠ける. その理由は, 当時のコンピュータは“オーバーフロー・アンダーフロー問題”, “算術演算処理が遅い”などの計算能力の低さにあった. 必然的に2項確率の近似確率計算法が多く開発され始めた. その当時, 数学の分野では既に正規分布やポアソン分布の確率計算に関しては既に豊富な計算法が確立されていたので, 2項確率の計算法として, 正規近似, ポアソン近似, グラム-シャリエ近似, その他多くの近似法が利用されるようになった. その中のいくつかは現在でも多用されているが, それらによる2項確率の近似値は, n が小さいときは無視できない程度の誤差が生じるという欠点を持っている. そのため, あまり大きくない n に対して近似精度をよくするための改良, 工夫が多く研究者によってなされてきた(例えば, Raff (1956), Bolshev, Gladkov and Shcheglova (1961), D’Ortenzio (1965), Pratt (1968), Peizer and Pratt (1968), Molenaar (1973), Ghosh (1980)). これら諸近似式について理論的な解説は, 例えば Johnson, Kotz and Kemp (1997), Patel and Read (1996), 竹内・藤野 (1981) に詳細に述べられている. 一般に, 離散型確率分布の確率関数や分布関数の値を直接計算することと, それらの値を正規近似等の近似式を使って計算することの決定的違いは, 前者はコンピュータの算術・論理演算装置に装備されている浮動小数点算術演算の誤差のみによるが, 後者はその誤差に加えて近似による誤差も生じるという点である. しかしながら, 現在でも数十程度の標本に対してさえ, 離散型確率分布の確率関数や分布関数の値の計算が直接確率計算法の代替的計算法である近似確率計算法で普通に行われている(例えば, 大野・磯田 (1993)). 直接確率計算が可能な現状において, なぜ代替的計算法が直接確率計算法に勝って使用されているのだろうか?

さらに, 多倍長技法も上で述べたコンピュータの弱点を補う長所を持っているため, この技法を実装している Maple や Mathematica も2項確率の直接確率計算が可能であり, 重用されている. しかし, これらのソフトウェアは何でも計算できて結果も瞬時に得られる訳ではない. 例えば, n の大きさとともに直接確率計算の計算時間も長くなり, 非常に大きな n に対しては, 2項確率の直接確率計算そのものが不可能となる. 多くの統計ユーザーおよび組織にとってはこれらのソフトウェアは高価であること, 推定, 検定の際に使用する確率値は相対誤差 10^{-7} から 10^{-10}

程度の精度があれば十分であること、瞬時の算出が要求されることを考えると、既存のフリーまたは安価なソフトウェア(例えば、Borland C++ Compiler, Xcode 2, GNUC, Visual C++)で手軽に確率値を計算する環境を構築することは必然と思われる(岩崎(2000), 松尾(2000), 竹越・岩崎(2001)).

2項確率を直接計算する試みとして、Bowerman and Scheuer (1990) はオーバーフロー(またはアンダーフロー)を避けながら、2項確率の直接確率計算アルゴリズムを提案している。これは非常に特殊な方法であり、一般性はないが標本の大きさが1023の場合まで計算が可能である。コンピュータの処理速度の高速化と処理方法の工夫(例えば、本論で提案するアルゴリズム)により、直接確率計算可能な標本の大きさの上限がかなり大きく(例えば、数億にまで)なっている。このため近似式は、通常の場合、その代替的役割を終えようとしている。本節では、広い範囲(例えば数億まで)の標本の大きさに対して2項分布の確率関数の値の直接確率計算アルゴリズム(第3.1節参照)や2項確率の直接確率計算アルゴリズム(第3.2節参照)を提案し、それらの有用性を数値実験で検証する(第3.3節参照)。

本論で提案する2項確率の直接計算のプログラムの実行ファイルはWindows XP用とMac OS X用の2種類をホームページ(平井(2006))で公開しているの、統計ユーザは適当な実行ファイルをダウンロードすればよい。他にも次のような応用も考えられる。

(i) 応用統計の分野では、多くの場合、確率論や数理統計学の分野で得られている結果を利用している。例えば、2項モデルにおいて得られるわずかな数百個のデータに対しても、2項確率に基づく漸近理論を利用して2項確率の近似値を求めている。逆に、広い範囲の標本の大きさまで直接確率計算可能な2項確率の計算アルゴリズム(または実行ファイル)は理論から導出される結果の数値的検証に役に立つと思われる。

(ii) 2項確率による正規確率近似が可能となる(平井・中村(2004b))。これにより、正規確率を必要とする統計手法において、2項確率が有用となってくる。

3.1. 2項確率関数の計算アルゴリズム

この節では2項分布 $B(n, p)$ の確率関数 $b(x; n, p)$ の値の直接計算法としてモグラ技法を使った計算アルゴリズムを提案する。

本論では以下通じて $p + q = 1, p > 0, q > 0, 0 \leq n \leq D - 1$ を仮定する。実際の計算プログラムにおいては、 $b(x; n, p)$ の値をこの技法で計算するとき使用する変数は倍精度実数型で表せる範囲の数のみである。すなわち、 p, q, n は倍精度実数型の変数として取り扱われる。確率関数 $b(x; n, p) (= {}_nC_x p^x q^{n-x})$ の計算は2項係数 ${}_nC_x$ と確率のべき乗 $p^x q^{n-x}$ の計算からなる。階乗とべき乗の値を算出する数種類の関数は既に第2.2節で導入されている。2項係数の値はこれらの関数を用いて算出する。その方法として次の2種類の計算法を考える。

(i) データセットによる方法: $\mathbf{c1}(n, x) = \mathbf{mfact1}(n) \otimes (\mathbf{mfact1}(x) \otimes \mathbf{mfact1}(n - x))$.

(ii) 近似式による方法: $\mathbf{c2}(n, x) = \mathbf{mfact2}(n) \otimes (\mathbf{mfact2}(x) \otimes \mathbf{mfact2}(n - x))$.

確率積 $p^x q^{n-x}$ の計算方法については次の2種類の計算法を考える。ただし、 $0 < p < 1, p \neq 1/2$ とする。

(i) 高速計算による方法:

$$\mathbf{p1}(x; n, p) = \mathbf{mpow1}(< A(p), l(p) >, x) \otimes \mathbf{mpow1}(< A(q), l(q) >, n - x).$$

(ii) 対数による方法:

$$\begin{aligned} p2(x; n, p) &= \text{mpow}(< 10, 0 >, < A(x), I(x) > \otimes \text{mlog10}(< A(p), I(p) >) \\ &\quad \oplus < A(n-x), I(n-x) > \otimes \text{mlog10}(< A(q), I(q) >)). \end{aligned}$$

残りの $p = 1/2$ に対しては次の計算法を考える。

(iii) 高速計算による方法：

$$p1(x; n, 1/2) = < 1, 0 > \otimes \text{mpow1}(< 2, 0 >, n).$$

(iv) 対数による方法：

$$p2(x; n, 1/2) = < 1, 0 > \otimes \text{mpow}(< 10, 0 >, < A(n), I(n) > \otimes \text{mlog10}(< 2, 0 >)).$$

$b(x; n, p)$ の近似値(モグラ数)を求めるために4通りの計算法を考える。

方法 I : $b11(x; n, p) = c1(n, x) \otimes p1(x; n, p)$, 方法 II : $b12(x; n, p) = c1(n, x) \otimes p2(x; n, p)$,

方法 III : $b21(x; n, p) = c2(n, x) \otimes p1(x; n, p)$, 方法 IV : $b22(x; n, p) = c2(n, x) \otimes p2(x; n, p)$.

方法 I ~ IV のどれが良いかを比較・検討するために次の数値実験をおこなう。真値 $t (\neq 0)$ に対する計算値 $\hat{t}$ の相対誤差を $|(\hat{t} - t)/t|$ で定義する (Dennis and Schnabel (1996))。本論では、比較の基準として相対誤差と計算時間を考える。数値実験の具体的な結果は平井・中村 (2004a) に詳細に述べてある。さらに詳細な計算結果やグラフ表示はホームページ (平井 (2006)) で公開している。ここでは、それらの一部の実験結果の概略を述べる。

数値実験 1 方法 I から IV を用いて、2項確率関数 $b(x; n, p)$ の値を計算し、その精度や計算時間を比べる。 p が $10^{-6}, 0.1, 0.2, 0.3, 0.4, 0.5$ の場合と n が 100, 500, 1000, 1 万, 15 万, 125 万, 1025 万, 1 億 25 万, 2 億 5025 万の場合を組み合わせたそれぞれの場合において、方法 I から IV の各方法で9個の2項確率関数 $b(kn/10; n, p) (k = 1, 2, 3, \dots, 9)$ の値を計算し、各計算値の相対誤差及び計算時間(ミリ秒)を調べた。相対誤差は Maple で計算した2項確率関数の値を真値としている。ただし、 $n \leq 2 \times 10^6$ の範囲では Maple の厳密値計算で算出した値を、それ以上の n に対しては浮動小数点近似による方法で算出した値を真値として採用した。

数値実験 1 の結果 方法 I から方法 IV までに共通して言える点は、相対誤差は p の値による影響を受けないということである。 n を 100 から 2 億 5025 万まで増やしていくと、方法 I では、すべての p の中の最大相対誤差(n を固定して p を動かしたときの相対誤差の最大値)のオーダーは 10^{-15} から 10^{-8} まで単調に増加する。同じ事を方法 II で行くと、その最大相対誤差は、各 n において方法 I のその 10 倍程度である。方法 III では、すべての p の中の最大相対誤差のオーダーは 10^{-8} から 10^{-13} ($n=1000$ のとき)まで減少した後、 10^{-6} まで増加する。方法 IV では、その最大相対誤差は、各 n において方法 III のその 10 倍未満である。計算時間については、方法 I では 0.08 ミリ秒から 107 ミリ秒まで単調に増加し、方法 II の場合も方法 I と大差ない。方法 III と方法 IV の場合を比較しても大差はなく、各 n において 0.08 ミリ秒から 0.16 ミリ秒までの僅かな増加(ほぼ一定)である。データセットを使う方法 I と II の比較では、相対誤差と計算時間の両方から判断して方法 II より方法 I の方が良い。さらに、近似式を使う方法 III と方法 IV の比較では、処理できる数の範囲の広さから方法 IV より方法 III の方が良い。方法 I と III を比較すると、最大相対誤差については、方法 I での最大相対誤差は方法 III でのその 10^{-7} 倍から 10^{-2} 倍となる。計算時間については、方法 III の方が短い、方法 I でも最大 107 ミリ秒程度なので、十分実用的である。

数値実験 1 の結果から、方法 I と方法 III が良い方法として残った。これらの方法では、さらに大きな n についての計算結果を調べ、モグラ技法がどの程度大きな n について計算可能であるか

を見るために次の数値実験をおこなう。

数値実験 2 p が $10^{-6}, 0.1, 0.2, 0.3, 0.4, 0.5$ の場合と n が 3 億, 4 億, 5 億, 10 億, 20 億, 50 億, 100 億, 200 億, 500 億の場合を組み合わせたそれぞれの場合において, $\bar{x} = \max\{0, [z_\alpha \sqrt{npq} + np - 0.5]\}$ に対して, 単項確率の値 $b(\bar{x}; n, p)$ を計算する. ここに, $[*]$ はガウス記号, z_α は標準正規分布の下側 $100\alpha\%$ 点である. ここでは, n と p の各組み合わせごとに, $\alpha = 0.01, 0.025, 0.05, 0.3, 0.5, 0.7, 0.9, 0.95, 0.975, 0.99$ の計 11 通りを考える. この実験での n の値では Maple による真値の直接計算ができないため, Prohorov の近似式を用いて Maple で計算した $b(\bar{x}; n, p)$ の値を真値として採用し, それぞれの値の相対誤差と計算時間を求める.

数値実験 2 の結果 まず, n が 5 億以下のときについて述べる. 方法 I では, n の増加による相対誤差の変化は小さく, むしろ p の値による変動の方が大きい. p を 10^{-6} から 0.4 まで変化させると, 最大相対誤差(p を固定して n を動かしたときの相対誤差の最大値)のオーダーは 10^{-2} から 10^{-5} まで単調に減少する. 同じ事を方法 IIIで行うと, その最大相対誤差は, 各 p において方法 I のその 10 倍未満である. $p=0.5$ のときは, 方法 I での最大相対誤差のオーダーは 10^{-8} であり, 方法 III での最大相対誤差は方法 I のその 10^2 倍程度である. 計算時間については全般的に方法 I の方が長い, n が 5 億のときでも最大 12 ミリ秒以下であるから実用的である.

次に, n が 5 億より大きい場合は, データセットが用意されていないため方法 I では計算ができない. したがって, 方法 III のみで考える. n を 10 億から 500 億まで増やしていくと, 最大相対誤差は p が極端に小さいとき ($p = 10^{-6}$) を除くと 10^{-4} 程度となるので, 方法 III も悪くはないと思われる.

以上の実験で, 計算結果の良さ具合の基準として, 相対誤差と計算時間を考えてきたが, 本論では, 確率計算を数回する場合を想定しているので, 計算時間より相対誤差を主な基準として選ぶことにする. 数値実験 1 と数値実験 2 の結果より, n が 3 億以下のときは方法 I が選ばれる. さらに, n が 3 億より大きく 5 億以下のとき, 相対誤差を主に判断する事, および既に n が 3 億以下のときは方法 I を採用している事を考慮して方法 I を選ぶ. n が 5 億より大きいときは, 方法 I にはデータセットがないので方法 III を選ぶことになる.

以上の議論より, モグラ技法を用いた 2 項確率関数の計算法として次を提案する.

- $n \leq 5$ 億のとき, 方法 I で算出された $b11(x; n, p)$ を $b(x; n, p)$ の値とする.
- $n > 5$ 億のとき, 方法 III で算出された $b21(x; n, p)$ を $b(x; n, p)$ の値とする.

(注 2) 計算時間を主に, 精度を従にした観点から判断するときは, $n \leq 5$ 億のときでも方法 III を採用するのも一案と思われる.

3.2. 2 項確率分布関数の計算アルゴリズム

本節では 2 項確率の値を数十～数百回求めることを前提としたときの 2 項確率を算出するアルゴリズムに焦点を当てる. 2 項確率は単項確率の和で表されている. この和の計算の実行過程においては精度または計算時間に関する問題点(誤差をどのようにして小さくするか, オーバーフロー・アンダーフローをいかにして避けるか)が考えられる. この節では, これらの問題を部分的に回避しながら 2 項確率 $B(x; n, p)$ の値を計算する種々のアルゴリズムを提案し, それらの良さ具合を比較・検討する. 以下通じて, 確率変数 X は 2 項分布 $B(n, p)$ に従うものとし, x は

$0 \leq x \leq n$ を満たす整数とする. 式 (4) の右辺の和の計算において, x が非常に大きい場合には, 計算する項の数も非常に大きなものとなるので計算時間の短縮方法を導入する. 現在のパソコンの数値精度は高々 15 桁ないし 16 桁であることを考慮して,

$$P(X \leq L) \leq 10^{-15}, \quad P(X \geq R) \leq 10^{-15} \quad (5)$$

を満たすような整数 $L = L(n, p)$, $R = R(n, p)$ ($L \leq R$) をまず決定する. その後, $x \leq L$ または $R \leq x$ なる x については式 (4) の右辺の和から除外する. 式 (4) の右辺の和の個数を大幅に減らすことにより, 計算時間がかかなり短縮される. 左側限界 L , 右側限界 R を求めるために上側および下側 2 項確率の評価が必要となる. ここでは, L, R を求めるために, 片側 2 項確率の評価を与える Kambo and Kotz が改良した Uspensky の不等式 (Kambo and Kotz (1966)) と Bahadur の不等式 (Bahadur (1960)) を利用した 2 種類の方法を考える. どちらの方法がよいかは数値実験の結果を基に議論する. 第 1 の方法として, 改良された Uspensky の不等式を利用した決め方を述べるために次の定理 (平井・中村 (2004a)) を用意する.

定理 2 確率変数は 2 項分布 $B(n, p)$ に従うものとする. 次の不等式が成立する.

$$P(X \geq n(q + \tilde{c})) \leq P(X \leq n(p - \tilde{c})) \leq 10^{-15}, 0 < p \leq 1/2. \quad (6)$$

$$P(X \leq n(q - \tilde{c})) \leq P(X \geq n(p + \tilde{c})) \leq 10^{-15}, 1/2 < p < 1. \quad (7)$$

ここに

$$\tilde{c} = \frac{1}{n} \sqrt{\frac{(180 \log 10)npq}{3 + \sqrt{9 + \frac{960 \log 10}{n}(pq)^2}}}$$

左側限界 $L_U(n, p)$, 右側限界 $R_U(n, p)$ をそれぞれ次式で定義する.

$$L_U(n, p) = \begin{cases} [n(p - \tilde{c})], 0 < p \leq 1/2, \\ [n(q - \tilde{c})], 1/2 < p < 1, \end{cases} \quad R_U(n, p) = \begin{cases} [n(q + \tilde{c})] + 1, 0 < p \leq 1/2, \\ [n(p + \tilde{c})] + 1, 1/2 < p < 1. \end{cases}$$

定理 2 より, $L_U(n, p)$, $R_U(n, p)$ は不等式 (5) を満たすことがわかる.

第 2 の方法として, Bahadur の不等式を利用した方法を述べる.

補題 1 (Bahadur, 1960) 確率変数は 2 項分布 $B(n, p)$ に従うものとする. 次の不等式が成立する.

$$P(X \geq x) < \frac{(x+1)q}{(x+1) - (n+1)p} b(x; n, p), np < x < n \quad (8)$$

$$P(X \leq x) < \frac{(n+1-x)p}{(n+1)p - x} b(x; n, p), 0 < x < np. \quad (9)$$

Bahadur の不等式 (8), (9) を用いて左側限界, 右側限界の候補を求める場合, 改良された Uspensky の不等式を用いる方法のように明確な形として解き, 左側限界, 右側限界を決定することができない. ここでは, 逐次 2 分法を用いた方法をのべる. 最初に, 不等式 (9) を用いて, 左側限界の候補を求めよう.

$$g(x) = \frac{(n+1-x)p}{(n+1)p - x} b(x; n, p), \quad x = 0, 1, \dots, n$$

とおく. 不等式 $g(x) \leq 10^{-15}$ は明確な形で解けないので逐次 2 分法を用いる. $np < 2$ のとき, $L_B(n, p) = -1$ とする. $np \geq 2$ かつ $g([np] - 1) \leq 10^{-15}$ のとき, $L_B(n, p) = [np] - 1$ とする. 以下, $np \geq 2$ と $g([np] - 1) > 10^{-15}$ を仮定する. 逐次 2 分法を始めるときの初期値を $(u_0, v_0) = (0, [np] - 1)$ とおき, 次のよう計算ステップで実行する.

【ステップ 0】 $g(u_0) > 10^{-15}$ ならば, $l_B(n, p) = -1$ とする. $g(u_0) \leq 10^{-15}$ のとき, $s_1 = [(u_0 + v_0)/2]$ とおく. $g(s_1) \leq 10^{-15}$ ならば, $(u_1, v_1) = (s_1, v_0)$ とおく. $g(s_1) > 10^{-15}$ ならば, $(u_1, v_1) = (u_0, s_1)$ とおく.

【ステップ 1】 $v_j - u_j \leq 1$ ならば, $l_B(n, p) = u_j$ として終了する. $v_j - u_j > 1$ ならばステップ 2 へとぶ.

【ステップ 2】 $s_{j+1} = [(u_j + v_j)/2]$ とおく. $g(s_{j+1}) \leq 10^{-15}$ ならば, $(u_{j+1}, v_{j+1}) = (s_{j+1}, v_j)$ とおく. $g(s_{j+1}) > 10^{-15}$ ならば, $(u_{j+1}, v_{j+1}) = (u_j, s_{j+1})$ とおく. ステップ 1 にとぶ.

上の逐次 2 分法(ステップ 0 からステップ 2)で得られた $l_B(n, p)$ を $L_B(n, p)$ とする. 以上の議論より, 左側限界 $L_B(n, p)$ は

$$L_B(n, p) = \begin{cases} -1, & np \leq 1, \\ [np] - 1, & np \geq 2, g([np] - 1) \leq 10^{-15}, \\ l_B(n, p), & np \geq 2, g([np] - 1) > 10^{-15} \end{cases}$$

となる. 不等式 (8) を用いて, 上と同様な手順により, 右側限界 $R_U(n, p)$ を求めることができるがここでは省略する(平井・中村 (2004a)). 上記の 2 方法で求めた限界 $L_U = L_U(n, p), R_U = R_U(n, p)$ と限界 $L_B = L_B(n, p), R_B = R_B(n, p)$ について, 組 (L_U, R_U) あるいは (L_B, R_B) のどちらが優れているかについては一概には言えない. 詳しい評価については平井・中村 (2004a) に述べられている.

左側限界 $L = L(n, p)$ と右側限界 $R = R(n, p)$ が与えられたときに, 2 項確率 $B(x; n, p)$ の近似値 $\hat{B}(x; n, p)$ を算出するアルゴリズムとして, 確率関数の値の小さい方から順に加えていく方法(アルゴリズム I)と, 逆に値の大きい方から順に加える方法(アルゴリズム II)を用意する.

アルゴリズム I: $m = [(n+1)p]$ とおく.

(a) $0 \leq x \leq m$ の場合:

(a1) $0 \leq x \leq L$ のとき, $\hat{B}(x; n, p) = 0$ とする.

(a2) $L < x \leq m$ かつ $x - L \leq R - x$ のとき,

$$\hat{B}(x; n, p) = b(L+1; n, p)(1 + \alpha_{L+2} + \alpha_{L+2}\alpha_{L+3} + \cdots + \alpha_{L+2}\alpha_{L+3} \cdots \alpha_x)$$

とする. ここに, $\alpha_i = (n-i+1)/i \times p/q (i = 1, \cdots, n)$ である. $b(L+1; n, p)$ は第 3.1 節で採用されたアルゴリズムにより計算する. 上式の右辺の第 2 因子の和は左から順に加法を行う.

(a3) $L < x \leq m$ かつ $R - x < x - L$ のとき, 次の 3 種類の場合分けを考える.

(a31) $m < R - 1$ のとき,

$$\hat{B}(x; n, p) = 1 - (b(x+1; n, p)(1 + \alpha_{x+2} + \alpha_{x+2}\alpha_{x+3} + \cdots + \alpha_{x+2}\alpha_{x+3} \cdots \alpha_{m-1}) + b(R-1; n, p)(1 + \beta_{R-2} + \beta_{R-2}\beta_{R-3} + \cdots + \beta_{R-2}\beta_{R-3} \cdots \beta_m))$$

とする. ここに, α_i は (a2) におけるものと同じであり, $\beta_i = (i+1)/(n-i) \times q/p (i = 0, 1, \cdots, n-1)$ である. $b(x+1; n, p)$ および $b(R-1; n, p)$ は第 3.1 節で採用されたアルゴリズムにより計算する. 上式の右辺の第 2 因子および第 3 因子の和は左から順に加

法を行う。

(a32) $x < R - 1 \leq m$ のとき,

$$\hat{B}(x; n, p) = 1 - b(x+1; n, p)(1 + \alpha_{L+2} + \alpha_{L+2}\alpha_{L+3} + \cdots + \alpha_{L+2}\alpha_{L+3} \cdots \alpha_{R-1})$$

とする。ここに、 α_i は (a2) におけるものと同じである。 $b(x+1; n, p)$ は第3.1節で採用されたアルゴリズムにより計算する。上式の右辺の第2因子の和は左から順に加法を行う。

(a33) $R - 1 \leq x$ のとき, $\hat{B}(x; n, p) = 1$ とする。

(b) $m < x$ の場合: $\hat{B}(x; n, p) = 1 - \hat{B}(y-1; n, q)$ とする。ここに, $y = n - x$, $q = 1 - p$ 。

(注3) 上述のアルゴリズムのモグラ技法による表現は平井・中村 (2004a) に詳しく述べられている。

アルゴリズム II は, 和を確率関数の値の大きい方から順に加法を実行していく方法である。加法の順序以外はアルゴリズム I と同様の手順なのでここでは省略する。詳細な手順については平井・中村 (2004a) に述べられている。これら2種類のアルゴリズムは共に“2項確率関数×和”の形の計算と左側限界 $L(n, p)$, 右側限界 $R(n, p)$ の決め方が基本となっている。2項確率関数の計算アルゴリズムはすでに第3.1節で提案されているので, アルゴリズム I と II の違いは和の計算をそれぞれの項の小さい方から加えるか, 大きい方から加えるかの違いとなる。さらに, 左側限界と右側限界の決め方は両者共に共通で, 限界 L_U, R_U と限界 L_B, R_B の2種類を考える。これらを合わせて考えると, 次の4つの組み合わせが考えられる。例えば方法 I は, 限界 L_U, R_U を用いてアルゴリズム I で計算するという意味である。

方法 I: 限界 L_U, R_U , アルゴリズム I; 方法 II: 限界 L_U, R_U , アルゴリズム II;

方法 III: 限界 L_B, R_B , アルゴリズム I; 方法 IV: 限界 L_B, R_B , アルゴリズム II。

2項確率分布関数の計算方法として, これら4つの方法のどれが一番よいかということに関しては理論的な証明はできない。数値的な比較・検討が必要となるので, 次節で詳しく議論する。

3.3. 数値実験による2項確率の計算アルゴリズムの比較

2項確率 $B(x; n, p)$ の計算法として方法 I, II, III, IV を前節で導入した。本節ではこれらのうちのどの計算法がよいのかを調べるために数値実験を行なう。比較する基準として絶対誤差, 相対誤差, 計算時間を採用する。真値 $t (\neq 0)$ に対する計算値 $\hat{t}$ の絶対誤差, 相対誤差はそれぞれ $|t - \hat{t}|$, $|t - \hat{t}|/|t|$ である。方法 I, II, III, IV の良さ具合を比較するためのいろいろな数値実験やその詳細な数値結果は平井・中村 (2004a) に掲載されている。これらの数値実験や他の数値実験についての詳細な計算結果やグラフ表示はホームページ(平井 (2006))で公開している。その1つとして, いくつかの (n, p) の組に対して, $L_B \leq x \leq R_B$ となる100個の x について計算した $\hat{B}(x; n, p)$ のうち, $0.0001 \leq \hat{B}(x; n, p) \leq 0.9999$ を満たす $\hat{B}(x; n, p)$ の中での絶対誤差, 相対誤差の最大値と最小値, および $1 - \hat{B}(x; n, p)$ の相対誤差の最大値と最小値を計算する実験がある。この実験の結果, 方法 I と方法 II の間には絶対誤差, 相対誤差, 計算時間のどれについても差がなかった。方法 III と方法 IV についても同様であった。この実験結果および和の計算は値の小さい方から順に加えるという数値計算の常識に従って, 方法 I と方法 III に焦点を当てることにする。

方法 I と方法 III のどちらを採用すればよいかを調べるために, 3種類の数値実験を行なう。

数値実験3 p が 10^{-6} , 0.001, 0.01, 0.1, 0.5 の場合と n が 20, 50, 100, 200, 500, 1000, 2000, 5000,

1 万, 2 万, 5 万, 10 万, 20 万, 50 万, 100 万, 200 万, 500 万, 1 千万, 2 千万, 5 千万, 1 億, 1 億 5 千万, 2 億, 2 億 5 千万の場合を組み合わせたそれぞれの場合において, $\bar{x} = \max\{0, [z_\alpha \sqrt{npq} + np - 0.5]\}$ に対して, $B(\bar{x}; n, p)$ を方法 I と方法 III で計算する. ここに, $[*]$ はガウス記号, z_α は標準正規分布の下側 $\alpha\%$ 点である. 本実験では, $\alpha = 0.05$, $z_\alpha = -1.644853626951472$ とする. 得られた計算値の最大相対誤差 (n を固定して p を動かしたときの相対誤差の最大値) と最大計算時間を基準にして方法 I と方法 III を比較・検討する.

数値実験 3 の結果 方法 I では, n を 20 から 2 億 5 千万まで増やしていくと, 最大相対誤差のオーダーは 10^{-16} から 10^{-8} まで単調に増加する. 同じ事を方法 III で行くと, 最大相対誤差のオーダーは方法 I と差がない. 最大計算時間については, 方法 I では, n を 20 から 2 億 5 千万まで増やしていくと, 0.01 秒から 64.06 ミリ秒まではほぼ単調に増加する. 同じ事を方法 III で行くと, 最大計算時間は, 0.14 秒から 5265.94 ミリ秒まではほぼ単調に増加する. これより, 方法 I は方法 III より優れていると判断される.

方法 I とよく知られた 2 項確率の近似計算法 (ポアソン近似, Peizer-Pratt の近似) との優劣を精度と計算時間について比較するために次の数値実験を行う.

数値実験 4 与えられた p に対して, n として, 20, 50, 100, 200, 500, 1000, 2000, 5000, 1 万, 2 万, 5 万, 10 万, 20 万, 50 万, 100 万, 200 万, 500 万, 1 千万, 2 千万, 5 千万, 1 億, 1 億 5 千万, 2 億, 2 億 5 千万を考える. 数値実験 3 で用いた $\bar{x}$ に対して, $B(\bar{x}; n, p)$ を方法 I, Molenaar のポアソン近似 (Molenaar (1973)), Peizer-Pratt の近似 (Peizer and Pratt (1968), 山内 (1992)) で計算する. ここに, ポアソン近似と Peizer-Pratt の近似についてはモグラ技法ではなく, 通常の C 言語を用いて計算している.

数値実験 4 の結果

(i) $p = 10^{-6}$ の場合. 相対誤差では, Molenaar のポアソン近似が圧倒的によい (最大相対誤差のオーダーはほぼ 0 (コンピュータの計算上 0 と判断される程度の小さな数) から 10^{-14} までの範囲) が, 方法 I も実用的に十分使える (最大相対誤差のオーダーは 10^{-16} から 10^{-8} までの範囲). Peizer-Pratt の近似では相対誤差は大きくなる (最大相対誤差のオーダーは 10^{-6} から 10^{-2} の範囲で変化する). 計算時間では, Molenaar のポアソン近似が, 0.0015 ミリ秒から 0.6703 ミリ秒の範囲となり, 方法 I は, 0.01 ミリ秒から 0.12 ミリ秒の範囲となる. さらに, Peizer-Pratt の近似では最大 0.0032 ミリ秒となる. 方法 I は他の方法に比べて何十倍もの時間がかかるが, 高々 0.12 ミリ秒なので, 実用上の問題はない.

(ii) $p = 0.5$ の場合. 相対誤差では, n が 20 から 200 万までは方法 I が最も良く (オーダーは 10^{-13} 程度), それ以上では Peizer-Pratt の近似が良い (オーダーは 10^{-12} 程度), という結果になる. Molenaar のポアソン近似は相対誤差が大きすぎて実用的ではない. 計算時間では, Peizer-Pratt の近似が圧倒的に短く, 方法 I は他の方法に比べて何千倍もの時間がかかるが, 高々 46 ミリ秒なので, 実用上の問題はない.

(i), (ii) の結果からわかるように, p の大小に関わらず, 方法 I で算出した値の相対誤差のオーダーは高々 10^{-8} 程度となる. したがって, 実用的な p の範囲内では, p の値によらず, 方法 I が非常に良い精度をもつ.

(注 4) ポアソン近似については, モグラ技法ではなく通常の C 言語を用いて計算してい

るため、 $p = 0.5$ の場合は、 n が 1000 を越えるとオーバーフローして計算不可能になったので、それ以上の n については数値実験ができなかった。

数値実験3と数値実験4より、方法IIIより方法Iの方が優れていると判断される。さらに、大きい n (>2 億 5 千万) といくつかの p の組に対して、方法Iの良さ具合を調べてみよう。

数値実験5 p が 0.1, 0.5 の場合と n が 3 億, 4 億, 5 億, 10 億, 20 億, 50 億, 100 億, 200 億, 500 億の場合を組み合わせたそれぞれの場合において、数値実験3で用いた $\bar{x}$ に対して、 $B(\bar{x}; n, p)$ を方法Iで計算する。ここでは、Peizer-Pratt の近似を用いて Maple で算出した値を真値として用いる。

数値実験5の結果 $p=0.1$ のとき、 n を 3 億から 500 億まで増やしていくと、最大相対誤差のオーダーは 10^{-8} から 10^{-4} まで単調に増加する。計算時間は、最大 142 ミリ秒である。 $p=0.5$ のとき、 n を 3 億から 500 億まで増やしていくと、最大相対誤差のオーダーは 10^{-8} から 10^{-5} までほぼ単調に増加する。計算時間は、最大 235 ミリ秒である。この程度の最大誤差と計算時間ならば実用上の支障はないであろう。

本節における各数値実験の結果に関する議論、および数十～数百回の確率計算を前提としてることより、 n が 500 億までは方法Iを2項確率の実用的な計算法として提案する。ここで注意すべき点は、モグラ算術を用いているので、何百億といった巨大な n に対しても方法Iの計算が可能であるということである。

(注5) 計算時間を主に、精度を従にした観点から判断するときは方法IIIを採用するのも一案と思われる。

4. おわりに

本稿では、パソコンとC言語による計算環境の下で、巨大数と微小数を処理するための算術システムを理論的に構築し、かつこの算術システムをパソコン上に実装(実現)した。この算術システムは巨大整数の計算技法として広く普及している多倍長技法とは異なる。算術システムという名から当然のことであるが、この算術システムは数値計算全般に適用可能である。応用として、特に2項確率 $B(x; n, p)$ の計算を取り上げ、このシステムを用いた計算アルゴリズムを提案した。本アルゴリズムの良さ具合をみるためにいろいろな数値実験を行った。よく知られているようにポアソン近似やPeizer-Prattの近似のどちらを使うかは、 n, p に強く依存している。これに対し、本アルゴリズムでは実用的な大きさ(例えば、数十、数百)の n に対して、その相対誤差は上記の2つの近似のそれより、 p が極端に小さい場合(例えば $p = 10^{-6}$)を除いて圧倒的によい。また、その相対誤差は n, p にはほとんど依存しない。従って、本アルゴリズムは広範囲の n と任意の p に対して十分実用的であると判断される。

従来、大きな標本数 n に対する2項確率の計算は、 n, p に関する条件によってポアソン近似、正規近似等の方法を使い分けて求めるのが標準的であった。本稿で提案するアルゴリズムにより、そのような近似式に頼ることなく n が数億まで実用的に2項確率の直接計算が可能となった。本アルゴリズムにより2項確率計算可能な n の範囲が押し上げられることになったが、 n が巨大(例

えば5億以上)なときは, 近似精度や計算時間から判断して, 本論で提案する計算技法を用いたポアソン近似や Peizer-Pratt の近似による確率計算が有用であると考えられる。

今回開発した2項確率計算プログラム(本論で提案する計算技法を用いたプログラム)は実行可能なファイルとしてホームページ(平井(2006))で公開している。

今後の課題としては, 2項分布だけでなく, 一般の確率分布(超幾何分布, 多変量2項分布等)のモグラ技法による確率計算システムを構築し, 公開することである。

謝 辞 本論文の改訂にあたり, 2人の査読者には表現や論理展開に関する適切な助言を頂き, また有用性の観点からも本論文の改善につながるとご教示を頂きました。編集理事の方々には多くの有意義なコメントを頂戴しました。ここに記して感謝の意を表します。

参 考 文 献

- Abramowitz, M. and Stegun, I.A. (Ed.) (1972): *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, Dover Publications, New York.
- Bahadur, R.R. (1960): Some approximations to the binomial distribution function. *Ann. Math. Statist.* **31**, 43–53.
- Bolshev, L.N., Gladkov, B.J. and Shcheglova, M.V. (1961): Tables for the calculation of B- and z-distribution functions. *Theory of Prob. Applic.* **6**, 410–419.
- Bowerman, P.N. and Scheuer, E.M. (1990): Calculation of the binomial survivor function. *IEEE Trans. Reliability* **39**, 162–166.
- Dennis, J.E., Jr. and Schnabel, R.B. (1996): Numerical methods for unconstrained optimization and nonlinear equations, SIAM, Philadelphia.
- D’Ortenzio, R.J. (1965): Approximating the normal distribution function. *System Design* **9**, 4–7.
- Gaul, W. (2005): Challenges concerning web mining. 2005 年度新領域融合プロジェクトによる研究会—大規模データ・リンケージ, データマイニングと統計手法—, 31–44. 統計数理研究所.
- Ghosh, B.K. (1980): Two normal approximations to the binomial distribution. *Commun. Statist.* **A9**, 427–438.
- 平井安久・中村 忠 (2004a): 巨大数・微小数を処理する算術システムとその2項確率計算への応用. *Technical Reports of Okayama Statistical Association* **90**.
- 平井安久・中村 忠 (2004b): 正規確率の2項近似について. 日本計算機統計学会第18大会論文集, 53–56.
- 平井安久 (2006): http://ed-www.ed.okayama-u.ac.jp/~suugaku/newhira/stat2006/mogura_webStart.htm.
- 石川真澄 (1999): 戦後政治史, 岩波書店.
- 岩崎学 (2000): 統計的推測における正確法と近似法. 応用統計学会チュートリアルセミナー, 1–26.
- Johnson, N.L., Kotz, S. and Kemp, A.W. (1992): *Univariate Discrete Distributions*, John Wiley & Sons, New York.
- Kambo, N.S. and Kotz, S. (1966): On exponential bounds for binomial probabilities. *Ann. Inst. Statist. Math.* **18**, 277–287.
- Knuth, D.E. (1981): *The Art of Computer Programming. Volume 2: Seminumerical Algorithms (2nd edition)*, Addison-Wesley, Reading, AM.
- 松本 眞 (2002): コイン投げで儲かる: 疑似乱数と符号理論. 数理科学 **29**, 8–13.
- 松尾精彦 (2000): 2次元統計量を用いた線形ロジスティックモデルの条件付正確検定. 応用統計学 **29**, 1–25.
- Molenaar, W. (1973): Approximations to the Poisson, binomial and hypergeometric distribution functions. *Mathematisch Centrum*.
- 森口繁一・宇田川鈺久・一松 信 (1995): 岩波 数学公式 III—特殊関数—, 岩波書店.
- 中沢信午 (1978): メンデルの発見, 共立出版.
- 大野 豊・磯田和男 (監修) (1993): 数値計算ハンドブック. オーム社.
- Patel, J.K. and Read, C.B. (1996): *Handbook of the Normal Distribution*, Marcel Dekker, New York.
- Peizer, D.B. and Pratt, J.W. (1968): A normal approximation for binomial, F, beta, and other common, related tail probabilities, I. *J. Amer. Statist. Assoc.* **63**, 1416–1456.
- Pratt, J.W. (1968): A normal approximation for binomial, F, beta, and other common, related tail probabilities, II. *J. Amer. Statist. Assoc.* **63**, 1457–1483.
- Raff, M.S. (1956): On approximating the point binomial. *J. Amer. Statist. Assoc.* **51**, 293–303.
- Robertson, W.H. (1960): *Tables of the Binomial Distribution Function for Small Values of p*. Sandia Corp., Albuquerque,

New Mexico, U.S.A.

Stuart, A. and Ord, J.K. (1994): *Kendall's Advanced Theory of Statistics*, Vol. 1, Sixth Edition, London.

竹越香苗・岩崎学 (2001): 2 段階抽出による 2 つの二項確率の検定におけるサンプルサイズの推定と検出力の評価.
応用統計学 30, 91-106.

竹内 啓・藤野和建 (1981): 2 項分布とポアソン分布. 東京大学出版会.

Weintraub, S. (1963): *Tables of the Cumulative Binomial Probability Distribution for Small Values of p* , The Free Press of
Glencoe, New York; III.1.

山内二郎編 (1992): 統計数値表. 日本規格協会.

(2005 年 3 月 28 日受付 2006 年 6 月 23 日最終修正 6 月 23 日採択)

著者連絡先: [REDACTED] (平井 安久)
[REDACTED] (中村 忠)

A New Arithmetic and an Application to the Computation of Binomial Probability for Very Wide Range of Sample Size

Yasuhisa Hirai¹ and Tadashi Nakamura²

¹ Faculty of Education, Okayama University

² Department of Mathematics Information Science, Okayama University of Science

Abstract

To calculate the binomial probability when the sample size is large, many useful approximations without straightforward calculation have been proposed and is often used presently. Faster and more powerful computers are available in the market today that is very cheap. In the present circumstances, question arises that should approximation be used. Among the various kinds of arithmetic which can deal with very large/small number, the most popular one is the multiple-precision arithmetic. In the practical field, for example in order to test hypotheses, we want to get probability quickly without accuracy. In these fields, the multiple-precision arithmetic has demerits such as long calculation time and dependence on the memory size of the personal computer. To avoid these demerits, we propose a new arithmetic on the set of real numbers. Based on the proposed arithmetic, we construct an arithmetic system by C language, which works on the personal computer. This arithmetic system is quite different from the multiple-precision arithmetic. This is applicable to various fields in which numerical calculation is needed. Especially, we try to use this system to calculate the binomial probabilities, because computation of binomial probability by a modern personal computer still includes problems. We propose an algorithm to calculate the binomial probability without approximations and program this algorithm using the proposed arithmetic system. This program enables us to calculate the binomial probability for very large sample size. Numerical experiments show that, from the standpoint of both relative error and calculation time, the proposed algorithm is useful within a very large range of sample size.

Key words: algorithm, arithmetic, binomial probability, over flow, under flow

Corresponding author

██ (Yasuhisa Hirai)

██ (Tadashi Nakamura)

Received March 28, 2005; Received in final form June 23, 2006; Accepted June 23, 2006.