数式処理 *Bulletin of JSSAC (2009)*
Vol. 16, No. 2, pp. 37 - 53

特集記事＜統計科学と数式処理＞

統計量分布漸近展開導出のための対称式基底変換アルゴリズム

山口 幸司　　仁木 直人*

マツダ株式会社　　東京理科大学

概　要

Reduction of the intermediate swell, found in the transformation process from a product of power sums in n independent variables into an integer weighted sum of augmented symmetric functions (ASF's), is our main concern, especially in the interest of application to the distribution theory of statistics. The main idea is to exchange the reverse lexical order on the set of ASF's build in the existing transformation algorithm for the length and reverse lexical order, which enables a new design of sorting algorithm for use in combining like ASF terms and, in addition, earlier elimination of ASF terms of higher order in $1/n$ than required. Numerical experiments have shown a distinct advantage of the new version both on computing time and requisite amount of memory.

1　はじめに

統計量の分布を求めることは，検定や推定の基礎であり，統計の分野における基本的なテーマである．しかし，統計量を表す式が簡単で，母集団が正規分布である場合などを除き，統計量分布を表す正確な式が求められることは稀であり，ほとんどの場合は標本サイズなどに関する漸近展開（Edgeworth 展開など）による近似に頼らざるを得ない．しかし，高次の漸近展開をするためには，高次の統計量モーメントが必要であり，その導出には膨大な量の計算が必要である．特定の統計量と母集団の組み合わせに限っては，数式処理の利用によって，この問題を乗り越えられる [11] が，他の一般的な統計量に対して適用できるような，汎用的な方法ではない．

統計量モーメントの汎用的な導出方法としては，対称式が成すベクトル空間の基底変換による方法 [7] があり，実際に統計量分布研究に使われている ([4] など)．統計量を表す式を Power Sum Product (PSP) の荷重和で表し，それを Augmented Symmetric Function (ASF) の荷重和に変換 （P2A 変換）し，その期待値を母集団モーメントで書くという手順で，統計量モーメントを導出することができる．しかし，この方法にも，P2A 変換途中において計算量・必要メモリが爆発的に増大し，中間膨張を引き起こすという問題がある．

*niki@ms.kagu.tus.ac.jp

そこで，中間膨張を抑えた P2A 変換アルゴリズムを提案する．アルゴリズムに取り入れる主な対策は以下の 2 つである．

- 長さ優先全順序の導入と，ASF が生成される際に従う順序規則に即したソート
- 変換途中において生成される不要な高次項の早期削除

また，アルゴリズムを Gnu Common Lisp [9] 上に実現するとともに，統計分布論において実際に生じる計算に適用し，これらの対策による中間膨張抑制効果を検証する．

第 2 節では，確率分布論に馴染みのない読者のために，統計量分布の漸近展開の方法および統計量モーメント導出の必要性を述べた後，P2A 変換による統計量モーメントの導出方法と，P2A 変換の問題点である中間膨張ついて述べる．第 3 節では，2 つの中間膨張対策の詳細を述べ，具体的なアルゴリズム設計について議論する．第 4 節では，対策の効果検証する数値実験結果を報告し，結果に対する考察を行う．最後の第 5 節では，本稿のまとめを行うとともに，関連話題についても述べる．

2 モーメント統計量分布の漸近展開における中間膨張

まず対称式基底変換によるモーメントの導出，およびその結果を用いた Edgeworth 展開による分布の近似が適用可能である統計量の族を示し，Edgeworth 展開の概要を述べる．次に，統計量モーメント導出の汎用的な方法である対称式の基底変換による方法と，その問題点である中間膨張について述べる．なお，本論文では母集団が十分高次のモーメントを持つと仮定している．

2.1 モーメント統計量

Edgeworth 展開が適用できる主要な統計量のほとんどは，標本モーメントの滑らかな関数で表せるモーメント統計量である．モーメント統計量の例として，以下のようなものがある．

$$\begin{array}{llll} \text{標本平均} & \dfrac{1}{n}\sum_{i=1}^{n} x_i = m'_1, & \text{標本分散} & \dfrac{1}{n}\sum_{i=1}^{n} x_i^2 - \left(\dfrac{1}{n}\sum_{i=1}^{n} x_i\right)^2 = m'_2 - m'^2_1 \\ \text{標本歪度} & m_3/m_2^{3/2}, & \text{標本尖度} & m_4/m_2^2 \end{array} \tag{1}$$

ここに，x_i はそれぞれ独立に同一の母集団に従う確率変数の実現値，n はサンプルサイズであり，m'_r は原点周りの r 次標本モーメント，m_r は平均周りの r 次標本モーメントを表す．

モーメント統計量を (漸近的に) 標準化した確率変数

$$X_n = \frac{\sqrt{n}\,(T_n - \theta)}{\sigma}$$

のキュムラント κ_r は，以下の Cornish-Fisher assumption を満たすことが知られている [1].

$$\begin{aligned} \kappa_1 &= n^{-\frac{1}{2}}\kappa_{1,1} + n^{-\frac{3}{2}}\kappa_{1,3} + n^{-\frac{5}{2}}\kappa_{1,5} + n^{-\frac{7}{2}}\kappa_{1,7} + \cdots \\ \kappa_2 &= 1 + n^{-1}\kappa_{2,2} + n^{-2}\kappa_{2,4} + n^{-3}\kappa_{2,6} + n^{-4}\kappa_{2,8} + \cdots \\ \kappa_r &= n^{-\frac{r-2}{2}}\kappa_{r,r-2} + n^{-\frac{r}{2}}\kappa_{r,r} + n^{-\frac{r+2}{2}}\kappa_{r,r+2} + n^{-\frac{r+4}{2}}\kappa_{r,r+4} + \cdots \qquad (r \geq 3) \end{aligned} \tag{2}$$

ここに，κ_{rs} は，統計量と母集団によって決まる定数で，n によらない．

2.2 Edgeworth 展開

標本サイズ $n \to \infty$ のとき，Cornish-Fisher assumption を満たす統計量の分布は正規分布に漸近する．その極限である正規分布の周りでの漸近展開は Edgeworth 展開と呼ばれる．

母集団パラメータ θ を推定する統計量 T_n を標準化した確率変数 X_n の Edgeworth 展開は，式 (2) のキュムラント κ_r を用いて，以下のように書ける．

$$\begin{aligned} F(x) &= \Pr[X_n < x] \\ &\sim \Phi(x) - \phi(x)\Big[\Big\{\frac{1}{6}\kappa_3 H_2(x) + \kappa_1\Big\} + \Big\{\frac{1}{72}\kappa_3^2 H_5(x) \\ &\quad + \Big(\frac{1}{6}\kappa_1\kappa_3 + \frac{1}{24}\kappa_4\Big)H_3(x) + \Big(\frac{1}{2}\kappa_1^2 + \frac{1}{2}(\kappa_2 - 1)\Big)H_1(x)\Big\} + \cdots\Big] \qquad (3) \\ &\sim \Phi(x) - \phi(x)\Big[n^{-\frac{1}{2}}\Big\{\frac{1}{6}\kappa_{3,1}H_2(x) + \kappa_{1,1}\Big\} + n^{-1}\Big\{\frac{1}{72}\kappa_{3,1}^2 H_5(x) \\ &\quad + \Big(\frac{1}{6}\kappa_{1,1}\kappa_{3,1} + \frac{1}{24}\kappa_{4,2}\Big)H_3(x) + \Big(\frac{1}{2}\kappa_{1,1}^2 + \frac{1}{2}(\kappa_{2,2} - 1)\Big)H_1(x)\Big\}\Big] + O\Big(n^{-\frac{3}{2}}\Big) \end{aligned}$$

ただし，$\Phi(x)$，$\phi(x)$ はそれぞれ標準正規分布の分布関数と密度関数であり，$H_j(x)$ は j 次 Hermite 多項式である．

式 (2) および (3) より，Edgeworth 展開のオーダを $n^{-r/2}$ までとしたとき，r+2 次までのキュムラントの $n^{-r/2}$ の項までが必要であることがわかる．モーメントからキュムラントを求めることができるから，n に関して高次の Edgeworth 展開をするためには，高次統計量モーメントの高次項が必要であり，その導出には膨大な量の計算が必要である．

特定の統計量と母集団の組については，既存の数式処理システムに組み込まれた関数の利用によって，この問題を乗り越えられる場合がある．しかし，モーメント統計量一般に対して適用できるわけではないため，以下のような，より汎用的な方法 [3] が提案されている．

2.3 対称式を用いた統計量モーメントの導出

統計量モーメントの汎用的導出方法として，対称式が成すベクトル空間の基底変換による方法がある [7]．モーメント統計量が標本モーメントの滑らかな関数で表せ，標本モーメント m'_r, m_r が観測値 $x_1, x_2, \ldots, x_n$ の対称式であるため，この方法が適用できる．

一般に，モーメント棟計量は，観測値の添字の入換えに関して不変な “対称関数” ではあるが，対称式 (対称性を持つ $x_1, x_2, \ldots, x_n$ に関する多項式) ではない．そのため，まず Taylor 展開により，対称式を係数とする漸近展開に展開する必要がある．

たとえば，標本尖度 m_4/m_2^2 は，補助的な確率変数

$$U = \frac{\sqrt{n}\,(m_2 - \mu_2)}{\mu_2}, \qquad W = \frac{\sqrt{n}\,(m_4 - \mu_4)}{\mu_4} \qquad (4)$$

を導入することで，Taylor 展開により，

$$\begin{aligned} \frac{m_4}{m_2^2} &= \frac{\mu_4}{\mu_2^2}\Big(1 + \frac{W}{\sqrt{n}}\Big)\Big(1 + \frac{U}{\sqrt{n}}\Big)^{-2} \\ &= \frac{\mu_4}{\mu_2^2}\Big(1 + \frac{1}{\sqrt{n}}(W - 2U) + \frac{1}{n}(3U^2 - 2UW) + \cdots\Big) \qquad (5) \end{aligned}$$

のように $1/\sqrt{n}$ に関するべき級数に展開できる．$U^r W^s$ は対称式 (すなわち対称多項式) であるため，(5) も対称式であり，対称式の基底変換法が適用できる．

統計量モーメントの導出に用いる対称式は，Power Sum (PS) の積である Power Sum Product (PSP) と Augmented Symmetric Function (ASF) である．$x_1, x_2 \cdots x_n$ を独立変数としたとき，PSP の定義は以下である．

$$\begin{aligned} p[r_1\ r_2\ \cdots\ r_l] &= p[r_1]\,p[r_2]\cdots p[r_l] \\ &= \prod_{j=1}^{l}\sum_{i=1}^{n} x_i^{r_j} = \sum_{i_1,i_2,\cdots,i_l} x_{i_1}^{r_1} x_{i_2}^{r_2}\cdots x_{i_l}^{r_l} \end{aligned} \qquad (r_1,\ r_2,\ \cdots,\ r_l \geq 1) \tag{6}$$

ここで，

$$\sum_{i_1,i_2,\cdots,i_l} = \sum_{i_1}\sum_{i_2}\cdots\sum_{i_l}$$

であり，和の範囲を明示しないときは，暗黙のうちに 1 から n までをわたるものとする．

対称式の引数部分 $\boldsymbol{\pi} = r_1\, r_2 \cdots r_l$ をパーティション (分割) と呼び，$l = |\boldsymbol{\pi}|$ をパーティション $\boldsymbol{\pi}$ の “長さ” と言うことにする．同時に $\boldsymbol{\pi}$ が規定する対称式の “長さ” も $|\boldsymbol{\pi}|$ とする．つまり，長さが 1 の PSP は PS と同義である．なお，パーティションの要素は，降順 ($r_1 \geq r_2 \geq \cdots \geq r_l \geq 1$) に並べるのを通例とするが，必ずしも降順でない場合も含めることにする．

PSP により，任意の標本モーメントをその荷重和で表すことができる．たとえば，標本分散 m_2 は，

$$m_2 = \frac{1}{n}\sum_i (x_i - m_1')^2 = \frac{1}{n}\sum_i x_i^2 - \frac{1}{n^2}\Big(\sum_i x_i\Big)^2 = \frac{1}{n}\,p[2] - \frac{1}{n^2}\,p[1\ 1] \tag{7}$$

である．

次に，ASF は以下のように定義される．

$$a[r_1\ r_2\ \cdots\ r_l] = \sum_{i_1\neq\cdots\neq i_l} x_{i_1}^{r_1} x_{i_2}^{r_2}\cdots x_{i_l}^{r_l} \tag{8}$$

ただし，

$$\sum_{i_1\neq\cdots\neq i_l} = \sum_{i_1}\sum_{i_2\neq i_1}\cdots\sum_{i_l\neq i_1,\cdots,i_{l-1}}$$

であり，“添字が相異なる場合についてのみ和をとる” という点で PSP とは異なる．

ASF の重要な性質は，$x_1, x_2 \cdots x_n$ がそれぞれ独立に同一の母集団分布に従う確率変数であるとき，その期待値が

$$\mathrm{E}\,a[r_1\ r_2\ \cdots\ r_l] = \prod_{j=1}^{l}(n+1-j)\,\mu_{r_j}' \tag{9}$$

のように，母集団の原点周りのモーメント μ_r' ときれいに対応していることにある．

以上より，PSP の荷重和で表した統計量のベキ乗 (または，必要な次数までの，その Taylor 展開) を ASF の荷重和に変換（以後，P2A 変換と呼ぶ）し，その期待値をとって母集団モーメントで表せば，必要なオーダーまでの統計量モーメントまたはその n に関する漸近展開 (しばしば “近似モーメント” と呼ばれる) が求められる．

2.4 対称式の基底変換法

P2A 変換の基礎となる公式 [7] は以下の 2 式である．

$$p[r_1] = a[r_1] \tag{10}$$

$$\begin{aligned} a[r_1\ r_2\ \cdots\ r_l] \times p[s] = {} & a[r_1{+}s\ r_2\ \cdots\ r_l] + a[r_1\ r_2{+}s\ r_3\ \cdots\ r_l] \\ & + \cdots + a[r_1\ \cdots\ r_{l-1}\ r_l{+}s] + a[r_1\ \cdots\ r_l\ s] \end{aligned} \tag{11}$$

式 (10) より，PS はそのまま ASF である．以降，(11) のように ASF に PS を一つずつかけていくことで，P2A 変換をすることができる．

標本分散 m_2 の 1 次モーメントと 2 次モーメントを例に，統計量モーメントを導出するまでの手順を以下に示す．

1. 統計量を PSP の荷重和で表す　— (12)
2. P2A 変換を行う　— (13)
3. 期待値を母集団モーメントで書く　— (14)

$$m_2 = \frac{1}{n}\,p[2] - \frac{1}{n^2}\,p[1\ 1] \tag{12}$$

$$= \frac{1}{n}\,a[2] - \frac{1}{n^2}\,(a[2] + a[1\ 1]) \tag{13}$$

$$\mathrm{E}[m_2] = \frac{n-1}{n}\,\mu_2' - \frac{n-1}{n}\,\mu_1'^2 \tag{14}$$

2 次以上の統計量モーメントは，(12) を累乗して展開 (15) し，以降は手順 2 と 3 を同じように実行 (16), (17) すれば求められる．

$$m_2^2 = \frac{1}{n^2}\,p[2\ 2] - \frac{2}{n^3}\,p[2\ 1\ 1] + \frac{1}{n^4}\,p[1\ 1\ 1\ 1] \tag{15}$$

$$\begin{aligned} &= \frac{1}{n^2}\,(a[4] + a[2\ 2]) - \frac{2}{n^3}\,(a[4] + 2\,a[3\ 1] + a[2\ 2] + a[2\ 1\ 1]) \\ &\quad + \frac{1}{n^4}\,(a[4] + 4\,a[3\ 1] + 3\,a[2\ 2] + 6\,a[2\ 1\ 1] + a[1\ 1\ 1\ 1]) \end{aligned} \tag{16}$$

$$\begin{aligned} \mathrm{E}\,m_2^2 = {} & \frac{(n-1)^2}{n^3}\,\mu_4' - \frac{4(n-1)^2}{n^3}\,\mu_3'\mu_1' + \frac{(n-1)(n^2-2n+3)}{n^3}\,\mu_2'^2 \\ & - \frac{2(n-1)(n-2)(n-3)}{n^3}\,\mu_2'\mu_1'^2 + \frac{(n-1)(n-2)(n-3)}{n^3}\,\mu_1'^4 \end{aligned} \tag{17}$$

表 1: P2A 変換における中間膨張の大きさ

l_P	#A	#S
1	1	1
2	2	2
3	3	5
4	5	15
5	7	52
6	11	203
7	15	877
8	22	4,140
9	30	21,147
10	42	115,975
20	627	5.17×10^{13}
30	5604	8.47×10^{23}

2.5 基底変換における中間膨張

P2A 変換による高次の統計量モーメント導出には，P2A 変換途中おいて計算量・必要メモリが爆発的に増大し，中間膨張を引き起こすという問題がある．PSP の引数であるパーティションの長さを l_P とすると，P2A 変換後の最終的な ASF の項数 $\#A$ は，整数 l_P の分割数 $\pi(l_P)$ で評価され，良く知られているように $O(\exp\sqrt{l_P})$ のスピードで増大する．この負荷増大も無視できないが，必要な計算であるので，単純には減らしようがない．

問題なのは，工夫せずに変換を行ったときに生じる，凄まじい中間膨張である．表 1 の $\#S$ に示す値は，P2A 変換途中に生成される ASF の同類項を全くまとめず，最終段階までそのままにしたときに現れる項の総数である．つまり，中間膨張を抑えるには，P2A 変換途中に ASF の同類項を要領良くまとめる必要がある．

近似統計量分布導出という観点からは，$\#A$ として数えられている中に存在する，不要な ASF にも注目すべきである．すなわち，P2A 変換後の ASF の荷重和には，最終的に Edgeworth 展開に使われない高次項が含まれている可能性がある．

統計量モーメント計算に頻出する例を用い，P2A 変換過程の詳細を見よう．このとき，標本モーメントの関数のみを扱っていることから，PSP にはパーティションの和をべきとする $1/n$ の累乗が掛けられていることに注意すべきである．たとえば，

$$\begin{aligned}\frac{1}{n^4}\,p[1\ 1\ 1\ 1] &= \frac{1}{n^4}\,(a[1]\times p[1\ 1\ 1]) \\ &= \frac{1}{n^4}\,(a[2]+a[1\ 1])\times p[1\ 1] \\ &= \frac{1}{n^4}\,(a[3]+3\,a[2\ 1]+a[1\ 1\ 1])\times p[1] \end{aligned} \tag{18}$$

$$= \frac{1}{n^4}\,(a[4] + 4\,a[3\ 1] + 3\,a[2\ 2] + 6\,a[2\ 1\ 1] + a[1\ 1\ 1\ 1])$$

において，それぞれの項の期待値は，整数係数を無視すれば，

$$\mathrm{E}\Big(\frac{1}{n^4}a[4]\Big) = \frac{\mu'_4}{n^3},\quad \mathrm{E}\Big(\frac{1}{n^4}a[3\ 1]\Big) = \frac{(n-1)\,\mu'_3\mu'_1}{n^3},\quad \mathrm{E}\Big(\frac{1}{n^4}a[2\ 2]\Big) = \frac{(n-1)\,{\mu'_2}^2}{n^3}, \tag{19}$$

$$\mathrm{E}\Big(\frac{1}{n^4}a[2\ 1\ 1]\Big) = \frac{(n-1)(n-2)\,\mu'_2{\mu'_1}^2}{n^3},\quad \mathrm{E}\Big(\frac{1}{n^4}a[1\ 1\ 1\ 1]\Big) = \frac{(n-1)(n-2)(n-3)\,{\mu'_1}^4}{n^3} \tag{20}$$

である．漸近展開に必要な次数が $O(n^{-1})$ であった場合，(19) の各項は $O(n^{-2})$ 以上であるため，実は不要であったことになる．また，さらに (18) の変換過程をさかのぼると，$a[3]$ からは $a[4], a[3,1]$ しか生成されないため，すでに (18) の 3 行目の時点で $a[3]$ が不要であったということになる．

このように，Edgeworth 展開の次数が与えられていれば，最終的に不要な高次項を P2A 変換途中のなるべく早い段階で削除できることになり，中間膨張を押さえる効果が高いと考えられる．

3 中間膨張の抑制策

P2A 変換途中における中間膨張への対策として，これまで見てきた

- ASF 同類項の要領良いまとめ
- 変換途中において生成される不要な高次項の早期削除

の 2 つの実現をめざす．まず，比較的簡単な後者について考えることとして，前者については ASF 集合への “長さ優先全順序” の導入とその順序による ASF 生成順に適した同類項整理に分けて議論する．

3.1 不要高次項の早期削除

P2A 変換で生成されるが，当初の目的達成には一切使われない高次項は，2.4 で議論したように中間膨張の原因になる．例として用いた (18) について，不要高次項しか生じない ASF を順次削除すると，

$$\begin{aligned} p[1\ 1\ 1\ 1] &= (a[1] \times p[1\ 1\ 1]) \\ &= (a[2] + a[1\ 1]) \times p[1\ 1] \\ &= (\cancel{a[3]} + 3\,a[2\ 1] + a[1\ 1\ 1]) \times p[1] \\ &= (\cancel{3\,a[3\ 1]} + \cancel{3\,a[2\ 2]} + 6\,a[2\ 1\ 1] + a[1\ 1\ 1\ 1]) \end{aligned} \tag{21}$$

のように，長さの短い ASF から削除すべきことがわかる．最終的に必要な ASF 長の最小値（例では 3）から，P2A 変換の各段階で削除すべき ASF を以下で判定ことができる．

$$\text{(削除すべき ASF 長)} \;<\; \text{(変換後必要な ASF 長)} - \text{(未変換 PSP 長)} \tag{22}$$

当然のことながら，実際の変換アルゴリズムにおいては，不要な高次項を作って削除するのではなく，そもそも不要項を作らない設計を行って，計算量とメモリ使用量を減らす．

3.2 ASF 集合への長さ優先全順序の導入

P2A 変換の進行に伴う ASF の同類項増大が中間膨張を起こす主な原因であるから，変換途中で同類項を効率良くまとめる必要がある．

同類項整理のタイミングは，ASF の荷重和に PS をかけ，同類項が生成された直後である．ここで，P2A 変換途中の ASF 長に注目し，

1. 同類項は同長の ASF の間にしか存在しないことは自明であり，同類項の整理は同長の ASF 間だけで行えばよい．
2. 乗算則 (11) より，長さ l の ASF に PS をかけると，長さ l と $l+1$ の ASF のみが生成され，かつ，長さごとに分けるための計算オーバヘッドは無いに等しい．
3. 同長の ASF 間だけで同類項の整理を行うことは，全ての ASF を対象にした整理よりも明らかに速い．
4. 長さごとの集合に分けて記憶すれば，不要な高次項の削除判断を一括で行える．
5. 同長の ASF 集合単位に PS との積を計算して同類項整理を行えば，PS をかけた直後に生じる同類項大量発生による中間膨張のピークを低く抑えることができる．

を考えれば，ASF 集合に長さ優先全順序を導入し，この順序に基づく記憶管理およびソーティングを行うことは，中間膨張対策として大変有意義であることが予想される．

ここでは，長さ優先全順序として，“$\succ$” と書き，以下で定義される “長さ優先逆辞書式順序” (Length and Reverse Lexical ordering; LRL) を用いる．ただし，係数は無視し，全順序 “$\underset{\mathrm{RL}}{\succeq}$” は逆辞書式順序 (Reverse Lexical ordering; RL) を表す．

$$a[\boldsymbol{\pi}] \succ a[\boldsymbol{\rho}] \underset{\mathrm{def}}{\iff} |\boldsymbol{\pi}| > |\boldsymbol{\rho}| \ \vee \ (|\boldsymbol{\pi}| = |\boldsymbol{\rho}| \ \wedge \ \boldsymbol{\pi} \underset{\mathrm{RL}}{\succeq} \boldsymbol{\rho}) \tag{23}$$

なお，既存のアルゴリズム [6] では，全順序として RL

$$a[\boldsymbol{\pi}] \succ a[\boldsymbol{\rho}] \underset{\mathrm{def}}{\iff} \boldsymbol{\pi} \underset{\mathrm{RL}}{\succeq} \boldsymbol{\rho} \tag{24}$$

が使われている．

2 つの順序による違いを以下に例示する．

$$\begin{aligned} \mathrm{LRL}: \ p[4\,3\,2\,1] \ &= a[10] + a[9\,1] + a[8\,2] + 2\,a[7\,3] + 2\,a[6\,4] + a[5\,5] + a[7\,2\,1] \\ &\quad + a[6\,3\,1] + a[5\,4\,1] + a[5\,3\,2] + a[4\,4\,2] + a[4\,3\,3] + a[4\,3\,2\,1] \end{aligned} \tag{25}$$

$$\begin{aligned} \mathrm{RL}: \ p[4\,3\,2\,1] \ &= a[10] + a[9\,1] + a[8\,2] + 2\,a[7\,3] + a[7\,2\,1] + 2\,a[6\,4] + a[6\,3\,1] \\ &\quad + a[5\,5] + a[5\,4\,1] + a[5\,3\,2] + a[4\,4\,2] + a[4\,3\,3] + a[4\,3\,2\,1] \end{aligned} \tag{26}$$

また以降では，LRL および RL に従ったソーティングを，それぞれ “LRL ソート” および “RL ソート” と呼ぶ．

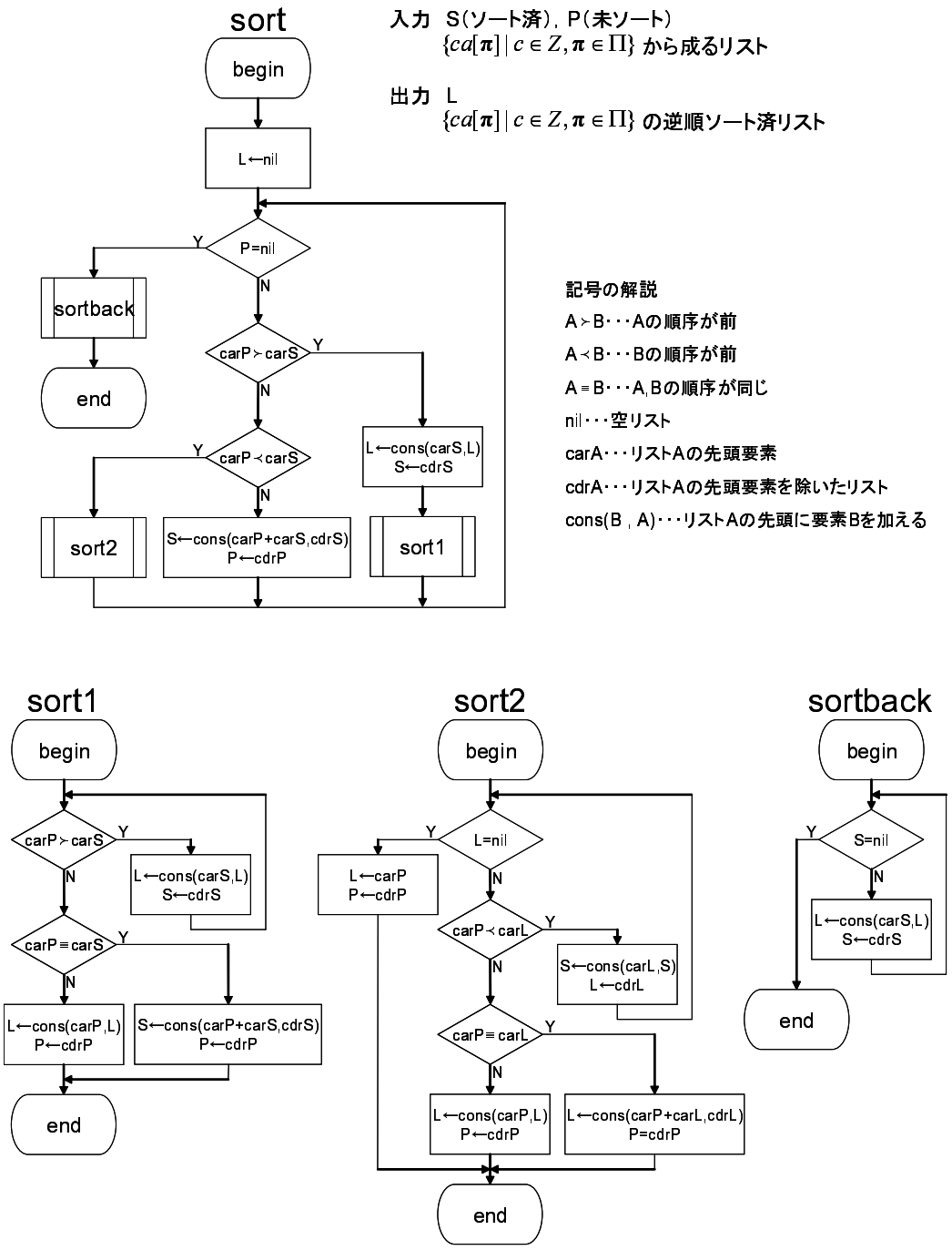

図 1: ASF リストの特徴を活かした LRL ソート・アルゴリズム

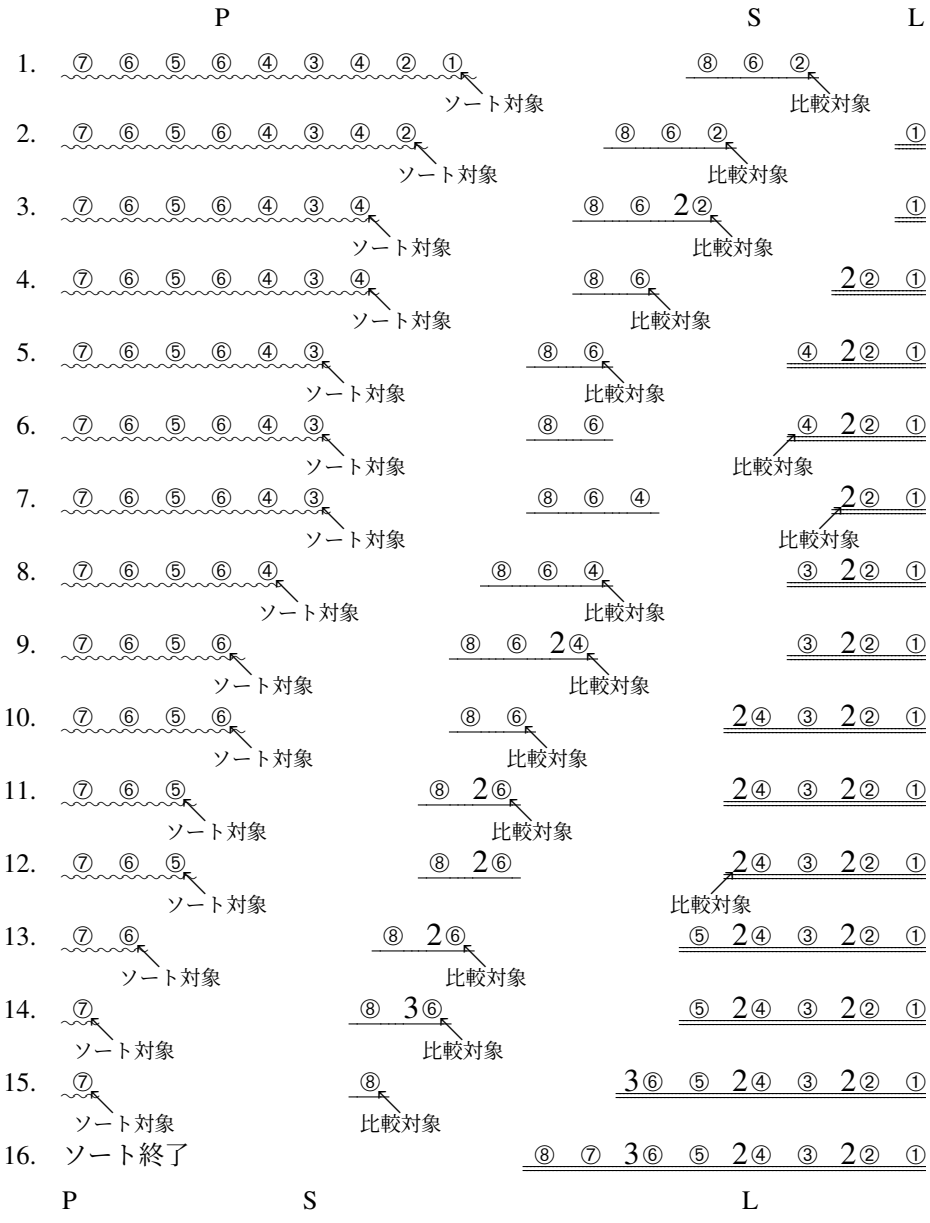

図 2: LRL ソートによる同類項整理の例

3.3 LRL ソートによる ASF 同類項の整理

同類項をまとめる際には，それぞれに特徴を持つ 2 つの ASF の集まりをソートする必要がある．例えば，

$$\Big(a[9\ 1] + a[7\ 3] + a[5\ 5] + a[7\ 2\ 1] + a[6\ 3\ 1] + a[5\ 4\ 1]\Big) \times p[1] \tag{27}$$

$$= a[10\ 1] + a[9\ 2]\ +\ a[8\ 3] + a[7\ 4]\ +\ a[6\ 5] + a[6\ 5] \tag{28}$$

$$+ a[9\ 1\ 1] + a[7\ 3\ 1] + a[5\ 5\ 1] \tag{29}$$

$$+ a[8\ 2\ 1] + a[7\ 3\ 1] + a[7\ 2\ 2]\ +\ a[7\ 3\ 1] + a[6\ 4\ 1] + a[6\ 3\ 2] \\ +\ a[6\ 4\ 1] + a[5\ 5\ 1] + a[5\ 4\ 2] \tag{30}$$

$$+ a[7\ 2\ 1\ 1] + a[6\ 3\ 1\ 1] + a[5\ 4\ 1\ 1] \tag{31}$$

において，(27) と比較すると，長さが変わらなかった (28), (30) と，長さが 1 増えた (29), (31) のあることが分かる．ここで，上の 各行中の ASF の並びをリストと呼ぶことにすると，

- 長さ l が変わらなかったリスト: l 個ごとに整列したグループの並びからなる．全体としてもある程度整列している．
- 長さが 1 増えたリスト: 初めから整列している．

という特徴を持つため，それを生かしたソーティング・アルゴリズムを用いる必要がある．

長さが変わらなかったリストを P，長さが 1 増えたリストを S，最終的に逆順ソートされて出力されるリストを L とし，メタ言語として Lisp で記述した LRL ソート・アルゴリズムのフローチャートを 図 1 に示す．また，式 (29) と (30) について，ソーティングによる同類項整理の進行状況を 図 2 に示す．ただし，図上において，各 ASF は丸数字で簡略化して表示され，リスト P (30) は波線部，リスト S (29) は下線部，リスト L は二重下線部で示されている．各リストの推移を見ると分かるように，S と L の間を双方向に挿入ソートするような動きをするアルゴリズムである．

4 効果の検証と考察

4.1 比較実験環境と測定項目

既存アルゴリズムと現実的な実行環境で比較するため，2 つの P2A 変換関数を Gnu Common Lisp を用いて作成した．LRL ソートを組み込んだ p2a-LRL と RL ソートを行う p2a-RL である．不要高次項の早期削除は，どちらのバージョンでも，実行・無効を簡単に切り替えられる．

計算に使った環境は，

CPU:	Intel Core 2 Duo E6600, 2.40GHz
Memory:	PC2-6400 (DDR2-800), 2 GB
Chipset:	Intel P965 (FSB 1066MHz)

であり，比較に用いる測定項目は以下のとおりである．

表 2: LRL, RL ソートを用いた P2A 変換比較

ソート	変換する PSP	real time (秒)	real-gbc (秒)	gbc time (秒)	gbc/real	memory (MB)
LRL	$p[11 \cdots 1]$	1.38	1.15	0.23	0.17	21.1
	$p[12\ 11 \cdots 1]$	15.07	12.58	2.49	0.17	50.5
	$p[13\ 12\ 11 \cdots 1]$	208.84	153.24	55.60	0.27	71.7
	$p[1^{30}]$	0.25	0.21	0.04	0.16	4.4
	$p[1^{35}]$	1.04	0.90	0.14	0.13	14.8
	$p[1^{40}]$	4.63	4.00	0.63	0.14	25.7
RL	$p[11 \cdots 1]$	8.07	6.53	1.54	0.19	50.5
	$p[12\ 11 \cdots 1]$	109.38	84.45	24.93	0.23	71.7
	$p[13\ 12\ 11 \cdots 1]$	2335.68	1440.23	895.45	0.38	97.3
	$p[1^{30}]$	4.12	3.61	0.51	0.12	37.2
	$p[1^{35}]$	32.20	27.48	4.72	0.15	50.5
	$p[1^{40}]$	228.53	189.34	39.19	0.17	71.7

real time: P2A 変換が終了するまでにかかった実時間（単位 秒）
gbc time: ガベージコレクションにかかった実時間（単位 秒）
real-gbc: (real time) − (gbc time) （P2A 変換自体にかかった実時間）
gbc/real: (gbc time)/(real time)
memory: メモリの割り当て容量 （メモリ使用量の目安．単位 MB）

また，それらより，いくつかの相対効率

real time 効率: (p2a-RL の real time)/(p2a-LRL の real time)
gbc time 効率: (p2a-RL の gbc time)/(p2a-LRL の gbc time)
real-gbc 効率: (p2a-RL の real-gbc)/(p2a-LRL の real-gbc)
memory 効率: (p2a-LRL の memory)/(p2a-RL の memory)

を計算している．

4.2 ソーティングの違いによる比較

LRL ソートの効果を検証するため，次の 2 パターンの PSP を P2A 変換し，RL ソートとの比較を行った．

- パターン A: 同類項があまり生成されない
  $p[11\ 10 \cdots 1]$, $p[12\ 11\ 10 \cdots 1]$, $p[13\ 12\ 11\ 10 \cdots 1]$
- パターン B: 同類項が大量に生成される（統計量分布計算で頻出する）
  $p[1^{30}]$, $p[1^{35}]$, $p[1^{40}]$

表 3: LRL ソートの相対効率（対 RL ソート）

変換する PSP	real time 効率	real-gbc 効率	gbc time 効率	memory 効率
$p[11\ \cdots\ 1]$	5.85	5.68	6.70	0.42
$p[12\ 11\ \cdots\ 1]$	7.26	6.71	10.01	0.70
$p[13\ 12\ 11\ \cdots\ 1]$	11.18	9.40	16.11	0.74
$p[1^{30}]$	16.48	17.19	12.75	0.12
$p[1^{35}]$	30.96	30.53	33.71	0.29
$p[1^{40}]$	49.36	47.34	62.21	0.36

計測実験結果を 表 2 に，また，RL ソート・アルゴリズムに対する LRL ソートの相対効率を 表 3 に示す．

表 2 より，LRL ソートの方が明らかに実行実時間が短く，メモリ使用量が少ない．パーティション長が大きくなった際の実時間の変化においても，RL ソートの増加傾向が甚だしい．表 3 からも，PSP 長が大きくなるにつれて，LRL ソートの相対効率が良くなっていくことが確認できる．すなわち，LRL ソートは大量の ASF が生成される変換であればあるほど，その使用効果が大きくなることが予想される．

パターン A とパターン B の場合を比較すると，同類項が大量に生成されるパターン B で，LRL ソートがより効果を発揮していることがわかる．たとえば，$p[1^{40}]$ の P2A 変換に至っては，実行実時間に約 50 倍もの差があり，より次数が上がれば，差がどんどん広がっていくと予想できる．また，メモリ使用量に関する効果についても，パターン A に比べてパターン B での使用効果が大きい．すなわち，統計量モーメントの導出時に直面する，同類項が大量に生成される場合には，実行時間とメモリ使用量のどちらの効果も非常に大きく，必要不可欠な対策であるといえよう．

4.3 不要な高次項の早期削除

ここでは，標本分散 m_2 の分布に関して，$O(n^{-10})$ までの高次 Edgeworth 展開に必要な統計量モーメントを実際に求めることで比較を行う．手順を以下に示す．

1. m_2 について，標準化を行う．

$$X_n = \frac{\sqrt{n}\,(m_2 - c_1)}{c_2} \tag{32}$$

   なお，母集団分散を σ^2 とするとき，$c_1 = \sigma^2, c_2 = \sqrt{2}\sigma^2$ であるが，これらは定数であるので c_1, c_2 としたまま計算を進める．

2. X_n の分布は Cornish-Fisher assumption を満たし，$O(n^{-10})$ までの Edgeworth 展開に必要なモーメントは 22 次である．したがって，必要な最高次モーメントは，

$$X_n^{22} = \frac{n^{11}}{c_2^{22}}\left(m_2^{22} + 22\,c_1 m_2^{21} + 231\,c_1^2 m_2^{20} + \cdots + c_1^{22}\right) \tag{33}$$

表 4: 不要高次項を早期削除する場合としない場合の比較

不要項	導出する m_2^r	real time (秒)	real-gbc (秒)	gbc time (秒)	gbc/real	memory (MB)
削除	m_2^{20}, $O(n^{-9})$	42.48	31.14	11.34	0.27	41.2
する	m_2^{21}, $O(n^{-\frac{19}{2}})$	85.62	59.68	25.94	0.30	64.0
	m_2^{22}, $O(n^{-10})$	179.70	115.27	64.43	0.36	63.2
削除	m_2^{20}, $O(n^{-9})$	91.91	60.31	31.60	0.34	58.3
せず	m_2^{21}, $O(n^{-\frac{19}{2}})$	205.37	120.50	84.87	0.41	89.3
	m_2^{22}, $O(n^{-10})$	450.50	238.57	211.93	0.47	116.6

表 5: 不要高次項早期削除の相対効率

導出する m_2^r	real time 効率	real-gbc 効率	gbc time 効率	memory 効率
m_2^{20}, $O(n^{-9})$	2.16	1.94	2.79	0.71
m_2^{21}, $O(n^{-\frac{19}{2}})$	2.40	2.02	3.27	0.72
m_2^{22}, $O(n^{-10})$	2.51	2.07	3.29	0.54

の期待値を求めることで得られる．これらの項の中で，最も激しい中間膨張発生が懸念される m_2^{22} は，(12) より，

$$m_2^{22} = \frac{1}{n^{22}}\, p[2^{22}] + \frac{22}{n^{23}}\, p[2^{21}1^2] + \frac{231}{n^{24}}\, p[2^{20}1^4] + \cdots + \frac{1}{n^{44}}\, p[1^{44}] \tag{34}$$

と PSP で表現できる．他のより低い次数の項についても同様である．

3. 全体にかかる係数 $n^{11}c_2^{-22}$ を乗じると，

$$\frac{n^{11}}{c_2^{22}}\, m_2^{22} = \frac{1}{c_2^{22}}\left(\frac{1}{n^{11}} p[2^{22}] + \frac{22}{n^{12}}\, p[2^{21}1^2] + \frac{231}{n^{13}}\, p[2^{20}1^4] + \cdots + \frac{1}{n^{33}}\, p[1^{44}]\right) \tag{35}$$

である．各項の n の次数を考慮すると，最終的に必要な次数が n^{-10} であることから，P2A 変換後に求めるべき ASF の長さは以下のように決定される．

$p[2^{22}]$: 長さ 1 以上
$p[2^{21}\,1^2]$: 長さ 2 以上
⋮
$p[1^{44}]$: 長さ 23 以上

これに従って、以降不要になる高次項を P2A 変換途中に削除することができる．

4. 式 (35) の各項について，途中削除する場合としない場合を，それぞれ LRL ソートを組み込んだ p2a-LRL を用いて計算する．また，求める統計量モーメントの次数による

効率の変化を比較するため，n^{-9} までの m_2^{20} と，$n^{-19/2}$ までの m_2^{21} を (35) と同様に計算する．測定項目は前と同様である．

不要な高次項を早期削除した場合としない場合の比較結を 表 4 に，および，高次項削除の相対効率を 表 5 にそれぞれ示す．

不要項削除は，表 1 における $\#A$ を減らすことを通じて $\#S$ も減らそうというものであり，直接 $\#S$ の削減を狙った改良版である p2a-LRL に付加しても，そもそも劇的に高い効果は期待できない．それでも，実用規模において 2 倍程度の高速化を実現し，メモリ使用量も顕著に低く抑えられている．また，次数が高くなるほど効率の上昇が見られることから，高次標本モーメント分布の高次モーメントが必要になる場合には，重要度が増すと考えられる．特に，メモリ使用量に対する効果が大きく，ガベジコレクション時間を大きく減少させていることに注目に値する．

5 おわりに

5.1 まとめ

標本サイズ $n \to \infty$ のとき，モーメント統計量分布が正規分布に収束する様子を記述するとともに，n がある程度以上大きいときに統計量分布を近似する Edgeworth 展開の導出には，高次の統計量モーメントを母集団モーメントで書く必要がある．その汎用的な方法である P2A 変換は，計算量・必要メモリの深刻な規模の中間膨張を起こしやすいため．その大幅な削減を目指して “LRL ソートの導入" と “不要高次項の早期削除" を提案した．

数値実験の結果，計算量・メモリ使用量の両面で，2 つの改善策は大きな効果を挙げることが確認できた．特に，対称式長が大きいとき，すなわち，標本モーメントの全次数が高く，かつ/または， Edgeworth 展開の n に関する次数が高いとき，より大きな相対効果を挙げることは実用上重要と思われる．

本研究では，アルゴリズムの改良に重点を置き，システム面での対策をあまり行っていない．たとえば，メモリ管理の効率化や，既存計算結果を利用した計算の省略，マルチコア・マルチスレッドを利用した並列化などである．また，実際的応用を考えれば，数式処理システムへのプラグイン化やパッケージとしての整備も必要であろう．

また，本稿で取り上げなかった 2 つの重要な関連話題，“重複度表現の利用"，“多変量統計量への拡張" について，以下で簡単に議論しておこう．

5.2 重複度表現の利用

実は，第 3 の改善策として，パーティションの重複度表現を利用して，不要な同類項の生成を抑制することも試みたが，そう大きな改善が見られなかった．参考のために，簡単に紹介しておこう．

重複度表現と数の並びによる通常の表現との違いとその違いが P2A 変換に与える影響は，以下の例を見れば容易に理解できよう．

- 通常表現に PS をかけた場合，一度同類項が生じ，それらをまとめる必要がある．

$$\begin{aligned} a[2\,2\,2\,1\,1\,1\,1]\times p[1] &= a[3\,2\,2\,1\,1\,1\,1]+a[3\,2\,2\,1\,1\,1\,1]+a[3\,2\,2\,1\,1\,1\,1] \\ &\quad + a[2\,2\,2\,2\,1\,1\,1]+a[2\,2\,2\,2\,1\,1\,1]+a[2\,2\,2\,2\,1\,1\,1] \\ &\quad + a[2\,2\,2\,2\,1\,1\,1]+a[2\,2\,2\,1\,1\,1\,1\,1] \\ &= 3\,a[3\,2\,2\,1\,1\,1\,1]+4\,a[2\,2\,2\,2\,1\,1\,1]+a[2\,2\,2\,1\,1\,1\,1\,1] \end{aligned} \tag{36}$$

- 重複度表現に PS をかけた場合，同類項の発生は少ない．

$$a[2^3\,1^4]\times p[1] = 3\,a[3^1\,2^2\,1^4]+4\,a[2^4\,1^3]+a[2^3\,1^5] \tag{37}$$

また，直ちに分かるように，重複度表現中のべきが概ね小さく，特に 1 の割合が多いときには，重複度表現処理に対する計算オーバヘッドによって，通常表現で変換するより効率が悪い．そのため，どのようなパーティションが多く現れるかをあらかじめ予測しておき，どちらの表現を用いるか選択する必要がある．

数値実験の結果は，実行時間・メモリ使用量ともに重複度表現と通常表現では大差がなく．アルゴリズムが複雑になることを考えれば，通常表現で十分という結論を得た．ただし，ここで用いたよりも大きな規模の対象に対する実験を行えば，重複度表現の採用を肯定する結論が出る可能性がある．また，重複度表現に向いたソート法やデータ形式などの工夫で，明確な効果が出ることも考えられる．

5.3 多変量統計量分布への拡張

パーティション中の数値をベクトルとすることで，P2A 変換を多変量統計量に適用することができる [3].

多変量版 LRL の定義がやや複雑になるになるものの，多変量 PS をかけた際の長さの変化は一変量の場合と同じであるため，同様な LRL ソートを適用することができる．不要な高次項の早期削除に関しても，同様に適用可能である．

ただし，重複度表現については，効果を期待することはできない．パーティション中の数値がベクトルに変わることで出現パターンが増え，重複度表現中の係数が大きくなりにくいためである．

参 考 文 献

[1] Bhattacharya, R. N. and Ghosh, J. K.: On the validity of the formal Edgeworth expansion. *Ann. Statist.*, **6**, 1978, 434–451.

[2] Graham, P. (著), 久野 雅樹, 須賀 哲夫 (訳): *ANSI Common Lisp*, ピアソン・エデュケーション, 2002.

[3] 中川 重和・仁木 直人: 対称式の変換アルゴリズムと その多変量統計分布論への応用. 計算機統計学, **4**(1), 1991, 35–43.

[4] Nakagawa, S., Niki, N. and Hashiguchi, H.: An omnibus test for normality. *Proc. 9th Jpn-China Symp. Statist.*, 2007, 191–194.

[5] 仁木 直人: 確率分布漸近展開の数式処理. 数学, 岩波書店, 38, 1986, 65–70.

[6] 仁木 直人: 対称式の計算パッケージ. 数式処理通信, **4**(2), 1986, 14–17.

[7] 仁木 直人: 対称式の計算と統計への応用. 数式処理通信, **4**(3), 1987, 6–10.

[8] Sagan, B. E.: *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*, Graduate Texts in Mathematics, Springer, 2001.

[9] Steele, G. L. (著), 井田 昌之 (訳): *COMMON LISP*, 共立出版, 1992.

[10] Stuart, A. and Ord, J. K.: *Kendall's Advanced Theory of Statistics, Volume 1*, *Distribution Theory*, Charles Griffin, 1987.

[11] Vrbik, J.: Finite-n Corrections to Normal Approximation. *Commun. Statist.–Simula. Comput.*, **34**, 2005, 827–837.