実用的な古典的誤差評価法の提案とGauss型積分公式の分点計算への応用について

幸谷智紀†

有限桁の浮動小数点演算を用いた数値計算によって得られる近似値は誤差を持つ．この誤差は理論誤差と丸め誤差から構成されているため，それぞれを個別に評価できれば，誤差はこれらの最大値として評価することができる．いわゆる「古典的誤差評価法」は，条件を変えて得られた複数の近似値から事後的にこれらの誤差を個別に評価する標準的な方法であり，既存の数値計算アルゴリズムを大幅に書き換える必要なく使用できるという利点を持つ．本稿ではこの古典的誤差評価法の考え方に基づき，ユーザが求める精度を持つ Gauss 型積分公式の分点計算が，2 つの異なる数値計算アルゴリズムに対してそれぞれ可能であることを示す．

On Proposal of Practical Classical Error Estimation Method and Its Application to Numerical Computations of Abscissas of Gauss Quadrature Rules

Tomonori Kouya†

Approximations obtained by numerical computations based on finite precision floating-point arithmetic lead to errors, which are primarily composed of theoretical and round-off errors. If we can estimate theoretical errors and round-off errors separately, the total error can be estimated as the maximum value of all the errors. The so-called “classical error estimation” (CEE) approach is a standard a posteriori method of estimating these errors; in this approach, errors are estimated by comparing several approximations obtained under various conditions. The advantage of this approach is that it is not necessary to rewrite the many existing numerical computation algorithms completely. In this paper, we demonstrate that our CEE-based error estimation method can be applied to two different algorithms in order to obtain the abscissas of Gauss quadrature rules with the user-required precision.

1. 初 め に

現在の数値計算の多くは有限桁の浮動小数点数演算と近似理論に基づいたものであり，それゆえに数値計算によって得られた結果は丸め誤差や理論誤差（打ち切り誤差）の影響を免れえない．したがって，数値結果に含まれる誤差の大きさを知る必要があるが，大規模計算においては目視で逐一チェックすることは難しい．そこで，数値結果に含まれる誤差の大きさを事後的に判断する手法がいくつか考案されている．特に近年では，自動証明手法としても使用できる厳密な精度保証を行うための有用な技術や理論が開発されているが，これも誤差を判断するための手法の 1 つといえる．しかし，今のところ，限られたケースを除き，既存の数値計算アルゴリズムを全面的に書き換える必要があり，厳密な誤差評価の必要のない計算では高性能化の妨げになりがちであり，ハイパフォーマンスな大規模計算を望むユーザには敷居が高い手法である．

これとは別に，かなり昔より「数値計算の常識」として浸透している誤差判定の手法[6] がある．これを簡単に述べると，数値計算結果に含まれる理論誤差および丸め誤差の大きさの評価は，条件を変えて複数回の計算を行い，その結果を比較することによって可能となる，というものである．理論的厳密さには欠けるが，既存の数値計算アルゴリズムに対しても簡単に適用できるという利点があり，現在でもごく標準的に用いられている．本稿ではこの手法を古典的誤差評価法と呼ぶことにする．

我々は古典的誤差評価法の考え方に基づき，次章で示すように，なるべく必要最小限の浮動小数点演算桁数で，ユーザが指定した精度を持つ近似値を自動的に

† 静岡理工科大学
Shizuoka Institute of Science and Technology

得るための実用的なアルゴリズムを提案する．

次にその適用例として，Gauss 型積分法の分点計算に適用した結果について報告する．これは三項漸化式で帰納的に表現される直交多項式の零点計算に相当するものであるが，1960 年代には 2 つの方式で計算することが提案されている．1 つは代数方程式の零点を Newton 法で計算するもので，日本では山下眞一郎が ALGOL プログラムとともに，分点と重みの数表を掲載した論文を執筆している．もう 1 つは，G.H.Golub と J.H.Welsch が提案したもので，三項漸化式を対称な実 3 重対角行列として表現し，その固有値問題として解く手法である．結論からいうと，直交多項式の分点数が上がるにつれて分点の近接度が増すため，前者は多倍長計算を用いなければ Newton 法の収束すらおぼつかなくなるのに対し，後者は固有値問題としては相当高い次数でも良条件であるため，使用した浮動小数点数の桁数に近い精度を持つ分点が得られるので，前者に比べて数値的に安定した手法であるといえる．しかし多倍長計算環境下であれば，前者でも桁数を任意に調整することで解くことは可能になるので，悪条件問題でも本稿で述べる手法が適用可能であることを示す実例としては望ましい．

そこで，本稿ではこの 2 つの手法に対して実用的な古典的誤差評価法に基づいた自動計算アルゴリズムが適用でき，ユーザが指定する精度を持つ Gauss 型積分公式の分点計算が可能であることを報告する．また，特に丸め誤差評価のための計算を並列化し，近年主流となりつつある Dual-core CPU マシン上において性能向上が図れることもあわせて示す．

2. 古典的誤差評価法に基づく理論誤差および丸め誤差評価

伊理ら[6] は，4 章「たった 1 回だけの計算なんて」（pp.20–25）において，「系統的に計算方法を変えて 2 回以上計算すると非常に有用な情報が得られる」と述べ，理論誤差と丸め誤差の大きさを評価する方法を具体例をもとに述べている．ことに後者は有限桁の浮動小数点演算を用いる限り逃れえないものであるが，厳密な丸め誤差限界を得ようとすると過大評価になりがちになる．しかし，実用的には同じアルゴリズムを計算桁数を変えて実行して得た近似値の差をとることで，短い計算桁数で計算した近似値に含まれる丸め誤差を評価することができる．

本稿ではこの考え方に基づき，10 進 S 桁の近似値 $x^S \in \mathbb{R}$ に含まれる理論（打ち切り）誤差 $T(x^S)$ と丸め誤差 $R(x^S)$ を下記のように評価する方法を「古典的誤差評価法」と呼ぶことにする．

理論誤差の評価 理論誤差が丸め誤差より優越している地点の誤差を理論誤差の評価値 $T(x^S)$ として使用

丸め誤差の評価 同じアルゴリズムを実行し，長い桁数 L（$> S$）で計算した結果 x^L を真値として用い，それより短い桁数 S による結果 x^S に含まれる丸め誤差の評価値 $R(x^S)$ を

$$R(x^S) = |x^L - x^S| \tag{1}$$

とする

本稿ではこれに基づき，x^S に含まれる絶対誤差の評価値 $E(x^S)$ を

$$E(x^S) = \max\left(T(x^S), R(x^S)\right) \tag{2}$$

とする．実際に適用するときには，$T(x^S) \approx R(x^S)$ となるような，必要最小限の桁数で計算することを仮定している．

上記の評価法のうち，丸め誤差の評価は問題やアルゴリズムによらず共通である．同じアルゴリズムを桁数を変えて実行するだけであるから，x^L の計算はあまり計算時間が変わらない程度に計算桁を増やし，x の計算と並列実行することが可能である．

これに対して理論誤差の評価法は，アルゴリズムが依拠する近似理論に応じたものが必要となる．今回取り上げるのは代数方程式の零点を求めるための Newton 法と固有値問題を解くための QR 法であるので，どちらも近似値が数列 $x_0, x_1, ..., x_n, ...$（$x_k \in \mathbb{R}$）という形で得られるタイプのアルゴリズムになっている．数列がどのように収束するかによって数列の値の変化も異なってくるため，すべての数値計算アルゴリズムや問題に適用可能な評価法を適用することは困難である．そこで，

- 各近似値 x_k の丸め誤差の評価値 $R(x_k)$ が得られている
- 数列は超 1 次単調に収束している

という仮定をおき，これに適合する場合のみ適用可能な理論誤差の評価法のみを考える．この場合は単調収束であるから，各近似値 x_k に含まれる理論誤差の評価値 $T(x_k)$ は

$$T(x_k) = |x_{k+1} - x_k| \tag{3}$$

としてよい．

以上の丸め誤差，理論誤差の評価値を用い，式 (2) に基づいて $E(x_k)$ を得る．これを x_k の絶対誤差として扱い，これに基づいて x_k の精度の判断を行う．重要なことは，これはあくまで精度の評価であって，解法の収束判定とは別の次元の話であるということである．したがって，収束判定の基準はアルゴリズムご

とに別途考える必要がある．

3. 要求精度を持つ近似値を得るための自動計算アルゴリズム

以上の古典的誤差評価法を用いて，ユーザが指定した精度桁数を持つ近似値を得るためには，計算桁数も当然それ以上必要となる．しかし，前述したように，計算桁数が多すぎると計算時間が長くなるため，必要最小限の計算桁数で実行することが望ましい．そこで，今回我々は，古典的誤差評価法によって得られる誤差の評価値に基づいて，下記のように文字どおりの Trial & Error を行うことで計算桁数を必要なだけ自動的に確保するようにした．この際の計算桁数の増減方法は，

(1) 収束するが要求精度より近似値の精度が低い場合 → 計算桁数の増量分を単調増加
(2) 収束しない場合 → 計算桁数の増量分を 2 倍に増加

とする．計算桁数さえ十分確保できれば収束する見通しがあれば，(2) の場合は計算桁数を早く増加させることで，失敗試行を減らすことができる．Ziv の戦略[10] と似ているが，これより計算桁数の増量は抑えた方が，今回対象とする Gauss 型積分の分点計算では少ない桁数で要求精度を満足することができるため，このように設定した．

以下，我々のアルゴリズムの詳細を述べる．

ユーザが指定した精度桁数（10 進）を $U \in \mathbb{N}$ とする．あらかじめ指定しておいた精度桁数の増分の最小値を $D \in \mathbb{N}$ を用い，桁数の増分量 C を

$$C = \max(D, U/10) \tag{4}$$

とする．つまり 10%増量する．これをもとに，実際に計算する桁数 S を

$$S = U + C \tag{5}$$

とし，丸め誤差を評価するために必要な精度桁数 L を

$$L = S + C \tag{6}$$

として，S より C 桁だけ余分に桁数をとるようにする．このようにして決めた精度桁数に基づいて $R(x^S)$ を求める．

さらに，収束判定に必要な相対許容度 ε_r と絶対許容度 ε_a を

$$\varepsilon_r = 10^{-U}, \quad \varepsilon_a = 10^{-2U} \tag{7}$$

とし，基本的には

$$|x_k^S - x_{k-1}^S| \le \varepsilon_r |x_k^S| + \varepsilon_a$$

を満足した時点で収束したものとする．このときの反復回数を k_s とすれば，この時点における理論誤差の評価値 $T(x_{k_s}^S)$ を得るために，さらにもう一度だけ反復を行い，$x_{k_s}^S$ の理論誤差の評価値 $T(x_{k_s}^S)$ を確定する．同時に丸め誤差の評価値 $R(x_{k_s}^S)$ も確定するので，式 (2) に基づいて絶対誤差の評価値 $E(x_{k_s}^S)$ が決定される．

もしこの時点で $E(x_{k_s}^S) < \varepsilon_r |x_{k_s}^S|$ を満足していれば，これをユーザ指定の精度を持つ近似値として受け入れる．そうでなければさらに C 桁追加してもう一度計算を行う．

これとは別に，もし指定された最大反復回数に達しても収束しない場合は，C を 2 倍して再計算を行うようにした．

今回実装した Gauss 型積分公式の分点計算は，以上の計算桁数増加アルゴリズムに基づき，ユーザ指定精度を満足するとともに，収束のために必要な桁数で自動的に計算が行えるようになっており，過大な桁数での計算を行わない．この実現のためには，次の 3 つの機能を持った多倍長浮動小数点ライブラリが不可欠である．

- 変数ごとに精度を可変に設定できる
- 異なる精度を持つ変数どうしの演算が可能
- プログラム実行中に動的に精度を変更する機能

我々の BNCpack[7] が土台としている GMP[1] や MPFR[9),11)] はこれらの機能を持っており，提案した自動計算アルゴリズムの実行が可能である．

実装と並列化

たとえば，ある近似値を得るための任意桁計算可能なサブルーチンとして，次のような引数をとり，停止時の反復回数を返す “num_algo” が提供されているものとする．このサブルーチンは普通の浮動小数点演算を前提としたアルゴリズムを実装したものでよい．

停止反復回数 k_s = num_algo(
近似値 x_{k_s},
初期値 x_0,
反復計算のための関数,
近似値の履歴 (と理論誤差評価値) $x_0, x_1, ..., x_{k_s}$, $T(x_{k_s})$,
相対停止条件閾値 ε_r,
絶対停止条件閾値 ε_a,
最大反復回数
)

通常の数値計算ではこのサブルーチンを一度だけ呼び出して実行する（**図 1** 上）が，我々が提案する丸め誤差の評価を行うためには，同じアルゴリズムを同じ停止条件，同じ初期値（ただし L 桁以上の精度が必要），同じ反復計算のための関数（ただし内部の計算は指定精度で実行）を与えた上で，S 桁と L 桁でそれぞれ実行し，近似値 ret_s($= x_{k_s}^S$) と ret_l($= x_{k_s}^L$)

表 1 3 項漸化式の係数，重み関数，積分区間と行列 T のタイプ
Table 1 Coefficients of three-term recurrences, weight functions, integral intervals and matrix types of T.

Name	$p_{-1}(x)$	$p_0(x)$	a_j	b_j	c_j	$w(x)$	a	b	Type of T
Legendre	0	1	$(2j-1)/j$	0	$(j-1)/j$	1	-1	1	Unsymmetric
Laguerre	0	1	$-1/j$	$(2j-1)/j$	$(j-1)/j$	$\exp(-x)$	0	$+\infty$	Symmetric
Hermite	0	1	2	0	$2j-2$	$\exp(-x^2)$	$-\infty$	$+\infty$	Unsymmetric

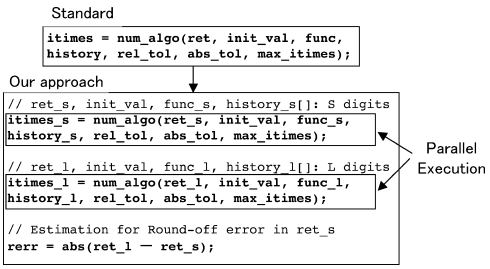

図 1 丸め誤差評価とその並列化
Fig. 1 Round-off error estimation and its parallelization.

を得る必要がある（図 1 下）．

しかし，この丸め誤差評価のためのサブルーチンの 2 重呼び出しの部分は同期をとる必要がないため，容易に並列化できる．今回我々はこの部分を POSIX Thread（Pthread）を用いて並列化した．このベンチマーク結果は後述する．

4. Gauss 型積分公式の分点計算と古典的誤差評価法の適用方法

Gauss 型積分公式の分点 $\alpha_1, \alpha_2, ..., \alpha_N$（$i < j$ のとき，$\alpha_i > \alpha_j$ とする）は，三項漸化式

$$p_j(x) = (a_j x + b_j)p_{j-1}(x) - c_j p_{j-2}(x) \quad (j = 1, 2, ..., N) \tag{8}$$

に基づいて定義される N 次直交多項式 $p_N(x)$ の零点

$$p_N(\alpha_i) = 0 \quad (i = 1, 2, ..., N)$$

として表現できる．ここで，a_j，b_j，$c_j \in \mathbb{R}$ は直交多項式ごとに定まる定数であり，今回計算した Legendre，Laguerre，Hermite 多項式の場合は**表 1** のようになる．

また，分点 α_i に対応する重み w_i（$i = 1, 2, ..., N$）は，$p_j(x)$ の最高次の係数を μ_j，$\lambda_j = \int_a^b (p_j(x))^2 dx$ とするとき，

$$w_i = \left(\frac{\mu_{N-1}}{\lambda_{N-1}\mu_N} p_{N-1}(\alpha_i) p'_N(\alpha_i) \right)^{-1} \tag{9}$$

となる．これによって，Gauss 型の数値積分は，重み関数を $w(x)$ とすると

$$\int_a^b w(x)f(x)dx \approx \sum_{i=0}^{N} w_i f(\alpha_i) \tag{10}$$

と計算される．今回計算した 3 つの多項式の場合は表 1 のようになる．

4.1 山下の方法

山下は，多倍長計算が可能な ALGOL 処理系を用いて Newton 法によって Legendre [13]，Laguerre [15]，Hermite [14] 多項式の零点を求め，重みを式 (9) に基づいた形で求めている．このとき，初期値は直交多項式ごとに適切なものを指定し，減次は行っていない．このようにして，山下は 2 点から 35 点までの Gauss-Legendre 積分公式の分点と重みを 10 進 20 桁の数表として提示している（Gauss-Laguerre は 26 点公式まで，Gauss-Hermite は 31 点公式まで）．使用した桁数は 30 点公式で 10 進 45 桁，Newton 法は 4〜5 回で収束していると山下は述べている．

この方法の欠点は，分点数 = 直交多項式の次数が増えるにつれて，零点の密集度が増し，きわめて悪条件の代数方程式問題となることである．したがって，もともと 10 進 20 桁の数表作成には多倍長計算が必要であるが，それ以上に，山下の提示した初期値（10 進 3 桁程度は正しい）を与えても，倍精度計算では 10 次程度でも Newton 法が収束しないという状況になる．また，収束したとしても精度は極端に落ちてしまうことになる．

なお現在は三項漸化式 (8) 式に基づき，係数を陽的に求めない方法で Gauss 型積分公式を求めるのが普通である [12]．今回はあえて悪条件になるよう，陽的に係数を導出して計算を行っていることをお断りしておく．

今回は後述する Golub らの方法による IEEE754 倍精度計算の結果（LAPACK の DSTERF ルーチンを使用）を初期値とし，Newton 法の計算はすべて多倍長計算で行った．Newton 法の最大反復回数は 100 回とし，この回数を超える反復を必要とする場合は桁数が足りないと判断，前述したアルゴリズムに基づいて自動的に桁数を増やして再計算するようにした．したがって，もっぱら収束に必要な桁数より大きめに計算桁数が設定されるため，誤差評価には理論誤差評価値が使用されることになる．

なお，後述するように Hermite 多項式の場合のみ，256 次を超える初期値を IEEE754 倍精度計算では得ることができないため，山下の方法では 256 次まで

の分点計算のみ実行している．また，重みの計算は式 (9) ではなく，後述する Golub らの方法と同様，連立一次方程式を用いている．

4.2 Golub&Welsch の方法

Golub ら[2] は，1 次から N 次までの式 (8) を

$$x\begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{N-2}(x) \\ p_{N-1}(x) \end{bmatrix} = \begin{bmatrix} -\frac{b_1}{a_1} & \frac{1}{a_1} & & & \\ \frac{c_2}{a_2} & -\frac{b_2}{a_2} & \frac{1}{a_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{c_{N-1}}{a_{N-1}} & -\frac{b_{N-1}}{a_{N-1}} & \frac{1}{a_{N-1}} \\ & & & \frac{c_N}{a_N} & -\frac{b_N}{a_N} \end{bmatrix} \times \begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{N-2}(x) \\ p_{N-1}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{a_N}p_N(x) \end{bmatrix} \tag{11}$$

と表現し，これを改めて

$x\mathbf{p}(x) = T\mathbf{p}(x) + \frac{1}{a_N}p_N(x)\mathbf{e}_N$（$\mathbf{e}_i$ は単位ベクトル）

とおき，分点 $x = \alpha_i$ においては

$$T\mathbf{p}(\alpha_i) = \alpha_i\mathbf{p}(\alpha_i) \tag{12}$$

となることを利用し，式 (12) を固有値問題のアルゴリズムを使用して解き，分点（T の固有値）を求めることを提案している．

現実の分点はすべて実数であるが，三重対角行列 T は非対称行列になることがある（表 1 参照）ので，

$$d_{i+1} = \sqrt{\frac{a_{i+1}}{a_i c_{i+1}}}\, d_i$$

を対角成分とする対角行列 D を用いて

$$J = DTD^{-1} = \begin{bmatrix} -\frac{b_1}{a_1} & \sqrt{\frac{c_2}{a_1 a_2}} & & & \\ \sqrt{\frac{c_2}{a_1 a_2}} & -\frac{b_2}{a_2} & \sqrt{\frac{c_3}{a_2 a_3}} & & \\ & \ddots & \ddots & \ddots & \\ & & \sqrt{\frac{c_{N-1}}{a_{N-2}a_{N-1}}} & -\frac{b_{N-1}}{a_{N-1}} & \sqrt{\frac{c_N}{a_{N-1}a_N}} \\ & & & \sqrt{\frac{c_N}{a_{N-1}a_N}} & -\frac{b_N}{a_N} \end{bmatrix} \tag{13}$$

と対称化し，この J を使用することも同時に提案している．ここで，重みはユークリッドノルムによって正規化された J の α_i に対応する固有ベクトル $\mathbf{q}_i$ の第 1 成分 $q_1^{(i)}$ を用いて

$$w_i = (q_1^{(i)})^2 \int_a^b w(x)dx \tag{14}$$

を計算することによって得られる．

今回はこの J に対して Implicit シフトを用いた実対称行列用の QR 法[3)~5)] を実行して分点を求めている．また，重みは分点 α_i を求めた後，連立一次方程式 $(J-\alpha_i I)\mathbf{q}_i = 0$ を解いて正規化し，式 (14) によって求めている．

本解法の利点は，代数方程式のような近接根が引き起こす問題が発生せず，固有値問題としてはきわめて良条件であるため，ほぼ計算桁数と同程度の精度を持つ分点を求めることができる点にある．しかし，三重対角行列を記憶しておくため，記憶領域も山下の方法に比べると多く必要になる．

誤差評価は，前述した山下の方法によるものと同様に行う．丸め誤差の評価のために S 桁計算で求めた行列 J^S と，L 桁計算で求めた行列 J^L それぞれに Implicit シフトつき QR 法を適用する．丸め誤差の評価および理論誤差の評価はそれぞれ特定の対角成分にのみ行う．

収束判定は J^S の副対角成分の値を見て収束を判断している[4),5)]．収束条件を満足していれば，もう 1 度反復を行って停止する．この際，精度の評価を行い，問題なければ減次して次の固有値の計算を行っている．このため，今回の問題のように数値的性質が良条件な行列の場合，前述したように理論誤差の評価値がゼロになりがちであるので，もっぱら丸め誤差評価値が使用されることになる．

なお，Hermite 多項式の場合のみ

$$d_{i+1} = d_1\sqrt{(2^i i!)^{-1}} \quad (i = 1, 2, ..., N-1) \tag{15}$$

となり，対称化する際に生じる対角行列 D の成分が急激に小さくなる．そのため，IEEE754 倍精度計算では 512，1024 次の D を陽的に求めることができない．しかし，今回使用した MPFR は指数部が 64 bit 環境下では 64 bit 長整数となるため，問題なく計算が可能となる．

5. 数値実験と分点の精度検証

数値実験を行った環境は以下のとおり，x86_64 Dual-core CPU マシンである．したがって，CPU 単体でも SMP マシンと同様の並列計算が可能である．

ハードウェア AMD Athlon64 X2 3800+，Fedora Core 4 x86_64

ソフトウェア gcc-4.1.1，GMP 4.2.1，MPFR 2.2.0，BNCpack 0.6c

Newton 法および QR 分解法は，BNCpack が提供している MPFR/GMP による多倍長演算をベースとしたデータ型および基本線型計算を用いて実装した．MPFR は変数ごとに仮数部の桁数を指定できるため，

表 2 分点 α_i の最大値と最小値
Table 2 Maximum and mimimum values of abscissas α_i.

	Gauss-Legendre		Gauss-Laguerre		Gauss-Hermite	
N	Maximum	Minimum	Maximum	Minimum	Maximum	Minimum
128	±9.998248879e − 1	±1.222369896e − 2	4.846155439e + 2	1.125138826e − 2	±1.529181976e + 1	±9.798382195e − 2
256	±9.999560500e − 1	±6.123912375e − 3	9.888402671e + 2	5.636640244e − 3	±2.199169337e + 1	±6.935239452e − 2
512	±9.999889909e − 1	±3.064962185e − 3	2.003068830e + 3	2.821067169e − 3	±3.143011738e + 1	±4.906344183e − 2
1024	±9.999972450e − 1	±1.533231356e − 3	4.038778564e + 3	1.411221668e − 3	±4.474456851e + 1	±3.470155326e − 2

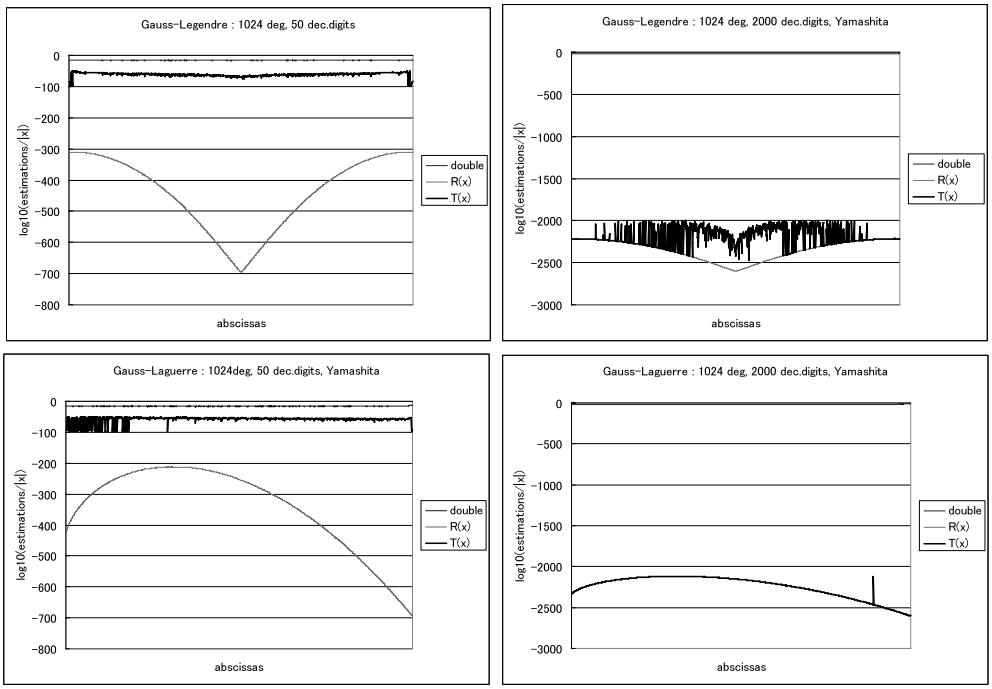

図 2 山下の方法における理論誤差，丸め誤差評価値：1024 次 10 進 50 桁（左），2000 桁（右），Legendre（上），Laguerre（下）
Fig. 2 Estimations for theoretical and round-off errors in Yamashita method: 1024th degree/50 decimal digits (left), 2000 digits (right), legendre (upper), laguerre (lower).

BNCpack のデータ型もすべて変数ごとに指定できるようになっている．また，BNCpack で提供している多倍長計算関数は，内部で実行されるすべての計算を返り値の桁数で実行するように実装されており，式 (1) に基づいた丸め誤差の評価値を容易に得ることができる．

以上の環境下で実際に得られた分点の値の最大値と最小値（絶対値の意味で）を**表 2** に 10 進 10 桁分のみ示す．

以下，

(1) 2 つのアルゴリズムの数値的性質
(2) 多項式の零点としての検証
(3) Gauss 型積分公式としての検証

を述べる．

5.1 2 つのアルゴリズムの数値的性質

1024 次の場合における山下の方法による，停止時における分点の近似値の理論誤差および丸め誤差の評価値のグラフを**図 2** に示す．また 256 次の Gauss-Hermite 公式の分点計算の評価値のグラフを**図 3** に示す．すべて，縦軸が（相対）誤差評価値の常用対数を示している．なお，評価値に対応する分点（横軸）は，$\alpha_1, \alpha_2, ..., \alpha_N$ の順に左から右に並べてある．

ユーザの要求桁数が低いにもかかわらず，直交多項式の計算における桁落ちが激しい（400 桁近い）ため，我々のプログラムでは試行を繰り返す．たとえば $N = 1024$, $U = 50$ のときの Gauss-Legendre 公式

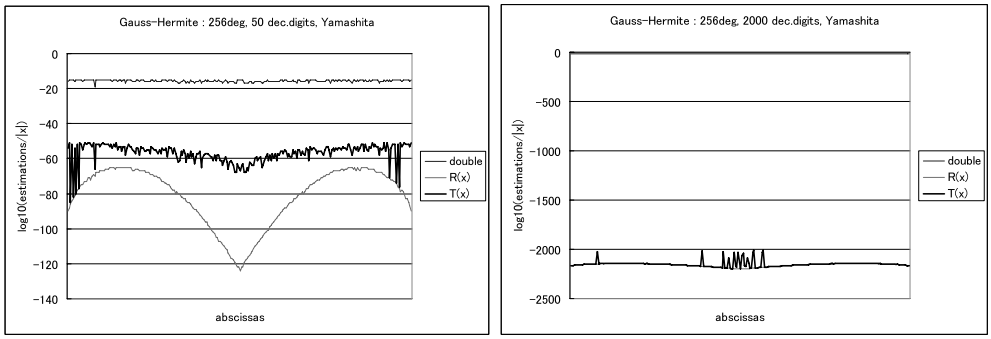

図 3 山下の方法における理論誤差，丸め誤差評価値（Hermite）：256 次 10 進 50 桁（左），2000 桁（右）
Fig. 3 Estimations for theoretical and round-off errors in Yamashita method (Hermite) : 256th degree/50 decimal digits (left), 2000 digits (right).

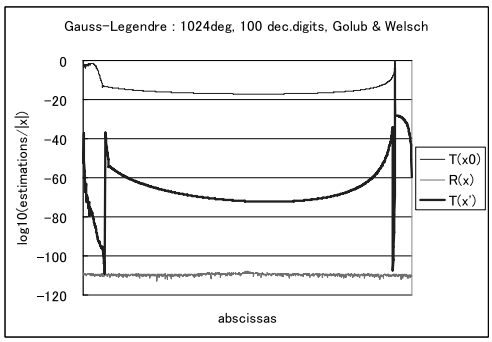

図 4 Golub らの方法における理論誤差，丸め誤差評価値：1024 次 10 進 100 桁
Fig. 4 Estimations for theoretical and round-off errors in Golub & Welsch method: 1024th degree/100 decimal digits.

の分点計算では，10 進桁換算で $S \approx 60 \to 81 \to 122 \to 203 \to 368 \to 695$ となる．また $U = 100$ の場合は，$S \approx 110 \to 131 \to 172 \to 253 \to 418 \to 745$ の精度が必要と我々のプログラムは判断している．しかし，要求桁数が上がるにつれて U と S の差は縮まっていき，$U = 1000$ のときは $S \approx 1100 \to 1300 \to 1701$，$U = 2000$ のときは $S \approx 2200 \to 2600$ で収束している．

なお，Golub らの方法の場合は，**図 4** に示すとおり，すべての精度桁において，丸め誤差の評価値は ε_r とほぼ一致し，理論誤差は初期評価値 $T(x_0)$ および停止 1 歩手前の評価値 $T(x')$ と順調に減少し，停止した時点では 0 となる．実際には理論誤差がゼロということはほとんどありえないが，前述したように，数値的に安定しているアルゴリズムを用い，問題がきわめて良条件の場合は，理論誤差が急激に丸め誤差以下に減少した後，$T(x)$ がゼロになることがある．このような場合でも，$E(x) = R(x) \neq 0$ となり，後述する Gauss 積分計算に示されるように，実際に得られた分点は十分な精度を得ており，今回実行した Golub らの方法は問題なく計算できている．

以上，山下の方法，Golub らの方法で得られた近似値は，10 進 5000 桁計算での結果と比較して相対誤差の検証を行ったところ，各要求精度〜+1 or +2 の精度を得ていることが確認できた．したがって，我々の自動計算アルゴリズムを適用することで，ユーザの要求精度に比して過大な精度桁を指定することなく，より短い計算時間でユーザの要求精度に近い最適な精度桁で近似値を得ることができているといえる．

5.2 多項式の零点としての検証

分点 α_i（$i = 1, 2, ..., N$）は直交多項式 $p_N(x)$ の零点である．ここではユーザ指定の精度 U で得られた分点の近似値を使って，絶対値最大の値を持つ $p_N(\alpha_1) = \sum_{i=0}^{N} \rho_i \alpha_1^i$ が 0 に近いこと，すなわち，$|\rho_M \alpha_1^M| = \max_i |\rho_i \alpha_1^i|$ に比べて絶対値のオーダが U に応じて小さくなることを確認する．

すべての値を表示するのは難しいため，$\log_{10}(|p_N(\alpha_1)|)$ のみを**表 4**，**表 5** に示す．この多項式の評価計算のみ 10 進 5000 桁計算で計算し，丸め誤差の影響を受けないように配慮してある．

このうち，Laguerre 多項式，Hermite 多項式の零点における多項式の値に極端な差が現れているが，これはすべての零点が $[-1, 1]$ 区間に含まれている Legendre 多項式に比べて零点の絶対値が大きいために起きている．実際，表 2 より，1024 次の Laguerre，Hermite 多項式の絶対値最大の零点はそれぞれ $\alpha_1 \approx 4.038778564 \times 10^3$，$4.474456851 \times 10^1$ となるため，多項式 $p_{1024}(\alpha_1) = \sum_{i=0}^{1024} \rho_i \alpha_1^i$ の各項の最大絶対値もそれぞれ $\log_{10} \max_i |\rho_i \alpha_1^i| \approx 1143.8$, 2043.7 となる（**表 3**）．したがって，桁数の増加に応じて多項式の値の絶対値が小さくなっているかどうか

表 3 直交多項式 $p_N(\alpha_1) = \sum_{i=0}^{N} \rho_i \alpha_1^i$ の絶対値最大項 $|\rho_M \alpha_1^M|$ とその係数 ρ_M
Table 3 Maximum absolute terms $|\rho_M \alpha_1^M|$ and the corresponding coeffients ρ_M of orthogonal polynomials $p_N(\alpha_1) = \sum_{i=0}^{N} \rho_i \alpha_1^i$.

	Legendre			Laguerre		
N	M	$\lvert\rho_M\rvert$	$\lvert\rho_M \alpha_1^M\rvert$	M	$\lvert\rho_M\rvert$	$\lvert\rho_M \alpha_1^M\rvert$
128	90	5.338360895e + 46	5.254880126e + 46	105	1.275544277e − 143	1.181452417e + 139
256	182	2.661968627e + 95	2.640760284e + 95	211	1.438311788e − 350	1.347315383e + 282
512	362	1.306096846e + 193	1.300902035e + 193	423	2.679397474e − 828	1.110174846e + 569
1024	724	6.257580545e + 388	6.245111719e + 388	847	2.106987733e − 1911	6.568225915e + 1143

	Hermite		
N	M	$\lvert\rho_M\rvert$	$\lvert\rho_M \alpha_1^M\rvert$
128	106	6.837431482e + 69	2.440984822e + 195
256	210	5.159612513e + 149	3.863119523e + 431
512	422	1.502008296e + 311	1.139424666e + 943
1024	846	1.522478264e + 647	5.115234205e + 2043

表 4 零点における多項式評価による検証（$\log_{10}(|p_N(\alpha_1)|)$）：山下の方法
Table 4 Verification by evaluating orthogonal polynomials at zeros ($\log_{10}(|p_N(\alpha_1)|)$): Yamashita method.

	Legendre				Laguerre				Hermite			
N	$U = 50$	100	1000	2000	50	100	1000	2000	50	100	1000	2000
128	−47.2	−96.7	−996.9	−1996.8	55.1	4.7	−895.3	−1894.5	127.2	78.5	−821.3	−1821.3
256	−47.3	−96.5	−996.7	−1996.5	164.7	113.9	−784.9	−1785.3	347.9	298.2	−601.9	−1602.2
512	−46.2	−95.9	−996.0	−1995.6	385.4	335.2	−567.0	−1564.8				
1024	−45.8	−96.1	−995.0	−1995.1	827.0	777.3	−122.6	−1122.7				

表 5 零点における多項式評価による検証（$\log_{10}(|p_N(\alpha_1)|)$）：Golub らの方法
Table 5 Verification by evaluating orthogonal polynomials at zeros ($\log_{10}(|p_N(\alpha_1)|)$): Golub & Welsch method.

	Legendre				Laguerre				Hermite			
N	$U = 50$	100	1000	2000	50	100	1000	2000	50	100	1000	2000
128	−47.2	−97.8	−996.9	−1996.8	55.1	4.7	−895.3	−1894.5	127.2	78.5	−821.3	−1821.3
256	−47.3	−96.5	−996.7	−1996.5	164.7	113.9	−784.9	−1785.3	347.9	298.2	−601.9	−1602.2
512	−46.2	−95.9	−996.0	−1995.6	384.9	335.2	−567.0	−1564.8	825.7	775.6	−124.2	−1123.9
1024	−45.9	−96.1	−995.0	−1995.1	827.0	777.3	−122.6	−1122.7	1859.3	1810.0	910.5	−89.5

で検証しなければならない．

実際，表 4，表 5 の値を比較すると，3 つの直交多項式すべてについて，$U = 50$ から $U = 100$ になった時点で約 50，$U = 100$ から $U = 1000$ になった時点で約 900，$U = 1000$ から $U = 2000$ になった時点で約 1000 ずつ $\log_{10}(|p_N(\alpha_1)|)$ が小さくなっていることが確認できる．よって，要求精度 U が大きくなるにつれて $p_N(\alpha_1) \to 0$ になっているといえる．

5.3 積分公式の精度の検証

得られた分点の精度を検証するため，次の定積分を Gauss 型積分公式 (10) で計算し，その相対誤差を計測した．

Gauss-Legendre $\int_0^{\pi/2} \cos x \; dx = 1$ (A)

Gauss-Laguerre $\int_0^{+\infty} \exp(-x) \cdot x \; dx = 1$ (B)

Gauss-Hermite $\int_{-\infty}^{+\infty} \exp(-x^2) \cdot \exp(x) dx = \exp(1/4) \cdot \sqrt{\pi}$ (C)

検証結果に及ぼす丸め誤差の影響を排除するため，式 (10) 右辺の積和計算のみ 10 進 5000 桁計算を行っている．この結果を**表 6**，**表 7** に示す．

なお，重み w_i は式 (9) で計算できるが，次数が増えると直交多項式の値が桁落ちにより不正確になる．そのため，山下の方法，Golub らの方法のどちらで求めた分点でも，重みは式 (14) に基づいて，すべてユーザの指定精度 U より 10%増やして計算し，U 桁に丸めたものを使用した．

表 6 および表 7 の結果より，山下の方法，Golub らの方法，いずれも分点および重みはすべて要求桁数の精度を保っており，過小評価（下線部分）になったところも，有効桁数 1 桁以内で収まっている．2 重下線の所は，山下の方法，Golub らの方法とも相対誤差の値が一致していることから，分点数が少ないことに起因する Gauss 型積分公式の理論誤差によるものと推察できる．

分点および重みの大部分は $U + C$ 以上の精度を

表 6 定積分計算による Gauss 型積分公式の検証（$\log_{10}$(相対誤差)）：山下の方法
Table 6 Verification of Gauss quadrature rule by definite integrals ($\log_{10}$(Relative Error)): Yamashita method.

	Gauss-Legendre (A)				Gauss-Laguerre (B)				Gauss-Hermite (C)			
N	$U=50$	100	1000	2000	50	100	1000	2000	50	100	1000	2000
128	−50.4	−99.8	−610.6	−610.6	−51.1	−101.6	−1000.8	−2001.1	−52.2	−100.9	−329.9	−329.9
256	−50.8	−100.4	−1001.5	−1374.1	−51.1	−102.0	−1000.6	−2001.5	−50.8	−101.4	−736.7	−736.7
512	−50.7	−100.8	−1002.6	−1999.9	−51.2	−101.5	−1001.8	−2001.6				
1024	−50.7	−100.6	−1000.1	−2000.3	−51.6	−102.3	−1001.1	−2001.4				

表 7 定積分計算による Gauss 型積分公式の検証（$\log_{10}$(相対誤差)）：Golub らの方法
Table 7 Verification of Gauss quadrature rule by definite integrals ($\log_{10}$(Relative Error)): Golub & Welsch method.

	Gauss-Legendre (A)				Gauss-Laguerre (B)				Gauss-Hermite (C)			
N	$U=50$	100	1000	2000	50	100	1000	2000	50	100	1000	2000
128	−52.0	−102.5	−610.6	−610.6	−50.6	−101.2	−1001.0	−2001.3	−50.4	−101.3	−329.9	−329.9
256	−52.0	−101.4	−1001.2	−1374.1	−51.1	−101.5	−1001.1	−2000.6	−50.7	−100.8	−736.7	−736.7
512	−51.7	−101.7	−1002.2	−2002.0	−51.5	−101.8	−1000.7	−2001.3	−50.5	−101.1	−1000.6	−1627.4
1024	−51.9	−101.7	−1002.0	−2001.8	−51.0	−101.2	−1001.1	−2001.6	−50.3	−101.3	−1000.5	−2000.6

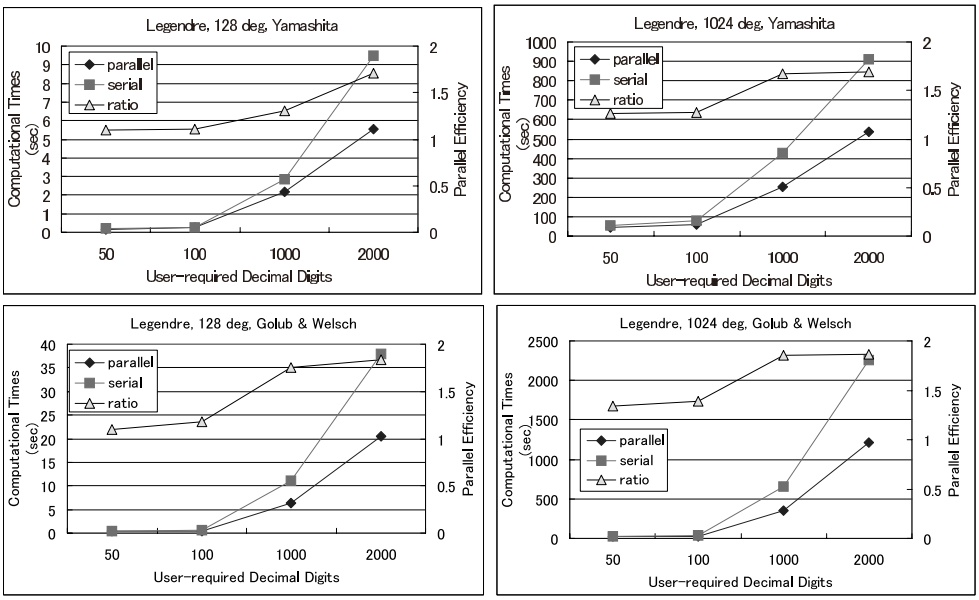

図 5 128（左），1024 次（右）Legendre 多項式の分点計算時間と並列化効率：山下の方法（上），Golub らの方法（下）
Fig. 5 Computational times and parallel efficiencies for 128th (left) and 1024th (right) degree Legendre orthogonal polynomials: Yamashita method (upper), Golub & Welsch method (lower).

持っていると思われるが，相対誤差がほぼ U と一致しているのは，近似値を U 桁（2 進変換するので $\lceil U/\log_{10} 2 \rceil$ bit）に丸めて格納しているためであろう．

5.4 計算時間と丸め誤差評価計算並列化の効果

今回数値実験に使用したのは Dual-core CPU であるので，丸め誤差評価のための計算を図 1 のように Pthread を用いて並列計算させることで，全体の計算時間を減らすことが可能である．今回は，山下の方法においては Newton 法の反復過程の，Golub らの方法においては 1 つの固有値を求めるための Implicit シフト付き QR 反復過程の丸め誤差評価のための計算を並列化した．こうして Pthread 化したプログラムを実行し，分点計算全体の計算時間と並列化効率（ratio）を計測した．今回実験した中で最小の 128 次と最大の

1024 次の Gauss-Legendre 公式の分点計算結果をプロットしたのが**図 5** である．

全体的に Golub らの方法に比べ，山下の方法の並列化効率が悪いことが分かる．これは並列化した部分の計算量が山下の方法の方が少ないためであろう．また，前述したように，代数方程式は次数が上がるにつれて悪条件になるため，U が小さい場合は計算桁数の変更数が多くなっていることも影響している．それでも 1024 次では要求桁数が多くなるにつれて，両者の並列化効率はともに上がっており，計算時間は十分短縮されている．他の 2 つの分点計算においても，計算時間および並列化効率はほぼ同じ傾向を示している．

なお，計算時間全体を比較すると，計算桁数の多い山下の方法の方が早くなっているが，これは初期値がほぼ 10 進 12〜16 桁の精度を持つ良好なものであることが，反復回数の減少につながっているためである．したがって，同じ初期値を QR 法のシフト値として採用できれば，Golub らの方法はさらにスピードアップできる可能性がある．

6. 結論と今後の課題

以上の検証結果より，我々の提案する古典的誤差評価法に基づいた Gauss 型積分公式の計算は，悪条件問題となる山下の方法でも，良条件である Golub らの方法でも，1024 次，要求精度 2000 桁まで有効に働くことが確認できた．また，丸め誤差評価のための計算を並列化することで，計算時間も短縮できることも確認できた．この結果をふまえ，今後は次のような展開を行っていきたい．

6.1 古典的誤差評価法の適用範囲の拡大

今回は数値積分によって得られた結果を検証することができる問題を選択したが，我々の提案する手法は，より広範囲のアルゴリズムおよび問題にも，理論誤差の評価が適切になされていれば適用可能であると予想できる．よって，零点計算や固有値計算以外のアルゴリズムにも逐次適用し，数値的な検証を行っていきたい．

また，丸め誤差の評価法については，今回はよりシャープな評価が得られる安全な方法を選択したが，同じ桁数で異なる丸めモードで計算した結果を比較して丸め誤差を評価する方法もある[8]．これとの比較検討も行っていきたい．

6.2 さらに高速かつ安定な任意精度 Gauss 型積分公式の計算プログラムの開発

今回は Legendre，Laguerre，Hermite 多項式の場合のみを計算したが，同様の方法により，他の直交多項式の場合も要求桁数の分点と重みを求めることは可能であると思われるので，複数の直交多項式に対応した Gauss 型積分公式の導出プログラムを開発していきたい．そのためには，多倍長計算を前提とした Newton 法および QR 法のスピードアップ技術が求められる．

現実問題としては高次の Gauss 型積分公式そのものにはそれほど実用価値はないと思われるが，ここで使用した Newton 法や QR 法で導出した分点は，前述したように，積和計算のみで済む定積分で精度の検証が実行できるというメリットがある．したがって，多倍長計算向けの代数方程式や固有値計算問題のためのテストベッドとしての利用価値は高いと，我々は考えている．

謝辞 本稿に対し，未知の査読者から再三にわたって的確な助言をいただいた．また，本稿で実行された計算は MPFR[11] の強力で正確な多倍長演算機能なくしては実現できなかった．関係者一同に感謝する．最後に，本稿全般にわたってアドバイスをいただいた永坂秀子先生に厚く御礼申し上げる．

参 考 文 献

1) Swox, A.B.: GNU MP. http://swox.com/gmp/
2) Golub, G.H. and Welsch, J.H.: Calculation of gauss quadrature rules, *Mathematics of Computations*, Vol.23, pp.221–230 (1969)
3) Golub, G.H. and van Loan, C.F.: *Matrix Computations* (3rd ed.), Johns Hopkins University Press (1996)
4) Stewart, G.W.: *Matrix Algorithms Volume I: Basic Decompositions*, SIAM (1998)
5) Stewart, G.W.: *Matrix Algorithms Volume II: Eigensystems*, SIAM (2001)
6) 伊理正夫，藤野和建：数値計算の常識，共立出版 (1985)
7) Kouya, T.: BNCpack. http://na-inet.jp/na/bnc/
8) 幸谷智紀，永坂秀子：IEEE754 規格を利用した丸め誤差の測定法について，日本応用数理学会論文誌，Vol.7, No.1, pp.79–89 (1997)
9) Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P. and Zimmermann, P.: MPFR: A multiple-precision binary floating-point library with correct rounding, *ACM Trans. Math. Softw.*, Vol.33, No.2, p.13 (2007)
10) Muller, J.-M.: *Elementary Functions, Algorithms and Implementation*, Birkhäuser (1997)
11) MPFR Project: The MPFR library. http://www.mpfr.org/
12) 杉原正顯，室田一雄：数値計算法の数理，岩波

書店 (1994)
13) 山下眞一郎：Gauss の数値積分公式の分点と重率の決定，情報処理，Vol.5, pp.206–215 (1964)
14) 山下眞一郎，佐竹誠也：Hermite-gauss の数値積分公式の分点と重率の決定，情報処理，Vol.5, pp.266–270 (1965)
15) 山下眞一郎，佐竹誠也：Laguerre-gauss の数値積分公式の分点と重率の決定，情報処理，Vol.4, pp.216–220 (1965)

(平成 19 年 4 月 23 日受付)
(平成 19 年 9 月 10 日採録)

幸谷 智紀（正会員）

1993 年日本大学大学院博士前期課程修了，同年石川職業能力開発短期大学校講師，1997 年日本大学大学院博士後期課程修了，1999 年静岡理工科大学講師，現在に至る．多倍長浮動小数点数を用いた高性能数値計算，誤差解析の研究に従事．SIAM，日本応用数理学会，日本数学会各会員．博士（理学）．