2007－HPC－111 (4)
2007／8／1

並列化した古典的誤差推定法に基づくGauss型積分公式導出プログラムの性能評価

幸谷智紀†

Gauss 型積分公式は直交多項式によって規定されるものであり，その分点は直交多項式の零点となる。この分点の精度は直接定積分の精度に影響を与えるため，高精度である必要がある。この零点を求める方法として，多倍長計算と Newton 法に基づいた山下の方法と，実対称行列の固有値問題に基づいた Golub & Welsch の方法の二つが 1960 年代に提案されている。本稿ではこの２つの方法に，いわゆる「古典的誤差推定法」を適用し，ユーザが指定した精度を持つ分点を求めることができることを示す。最後に，並列化した古典的誤差推定法を用いて，両者の計算速度の評価を行う。

Performance Evaluation of Programs to Calculate Gauss Quadrature Rules by Using Parallelized Classsical Error Estimation

TOMONORI KOUYA†

Gauss quadrature rules are defined by the corresponding orthogonal polynomials and its abscissas are the zero points of these polynomials. The zero points must be accurate because their accuracy affects one of the definite integrals computed by the Gauss quadrature rules. Two methods to obtain the abscissas have been proposed in the 1960's. One is proposed by Yamashita, based on the Newton method and multiple precision arithmetic. The other is proposed by Golub and Welsch, based on the eigenvalue problem of real symmetric matrix. In this paper, we apply the so-called "classical error estimation" to the two methods, and demonstrate that our proposed method can obtain the abscissas with the user-required precision. Finally, we evaluate the performance of two methods by using the parallelized classical error estimation.

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，それ故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) の影響を免れ得ない。従って，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。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。我々はこれを「古典的推定法」と呼ぶことにする。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及び丸め誤差の大きさの推定は，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。

我々はこの手法を，Gauss 型積分法の分点計算に適用し，任意の精度の公式を導出できることを示す。分点の計算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。一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 プログラムと共に，分点と重みの数表を掲載した論文を執筆している。もう一つは，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。結論から言うと，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言える。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事は可能になる。

本稿ではこの２つの手法によって，ユーザが指定す

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

る精度を持つ Gauss 型積分公式の導出を行った結果について報告する。また，特に丸め誤差推定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併せて示す。

2. 丸め誤差推定を並列化した古典的誤差推定法

本稿では，例えば伊理[6] に述べられているように，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 $R(x^S)$ を下記のように推定する方法を「古典的誤差推定法」と呼ぶことにする。

理論誤差の推定　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推定値 $T(x^S)$ として使用

丸め誤差の推定　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推定値 $E(x^S)$ を

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

とする。実際に適用する時には，$T(x^S) \approx R(x^S)$ 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。

上記の推定法のうち，丸め誤差の推定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 C 桁だけ増やし，x の計算と並列実行することが可能である。今回はこの部分を POSIX Thread(Pthread) を用いて並列化する。

これに対して理論誤差の推定法は，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，全ての数値計算アルゴリズムや問題に適用可能な推定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

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

として良い。

以上の丸め誤差，理論誤差の評価値を用い，(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推定であって，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。従って，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. 計算桁数の増加方法

以上の古典的誤差推定法を用いて，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，古典的誤差推定法によって得られる誤差の推定値に基づいて，下記のように文字通りの 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。

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。これを元に，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取るようにする。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。

更に，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。この時の反復回数を 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，更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，(2) 式に

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共に，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，過大すぎる桁数での計算を行わない。この実現のためには，変数ごとに精度を可変に設定できる多倍長浮動小数点演算ライブラリが必須となるが，我々の BNCpack[7] が土台としている GMP[1] や MPFR[8] はその機能を持っており，実行中に動的に計算桁数が変更できる柔軟性も兼ね備えている。

4. Gauss 型積分公式の分点計算

Gauss 型積分公式の分点 $\alpha_1, \alpha_2, ..., \alpha_N$ は，三項漸化式

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

に基づいて定義される N 次直交多項式 $p_N(x)$ の零点

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

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, $\lambda_j = \int_a^b (p_j(x))^2 dx$ とする時，

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

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

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

と計算される。

4.1 山下の方法

山下眞一郎は，多倍長計算が可能な ALGOL 処理系を用いて Newton 法によって Legendre[9], Laguerre[11], Hermite[10] 多項式の零点を求め，重みを (9) に基づいた形で求めている。この時，初期値は直交多項式ごとに適切なものを指定し，減次は行なっていない。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，極めて悪条件の代数方程式問題となることである。従って，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。

今回は後述する Golub & Welsch 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 法の計算は全て多倍長計算で行った。Newton 法の最大反復回数は 100 回とし，この回数を越える反復を必要とする場合は桁数が足りないと判断，自動的に桁数を増やして再計算するようにした。従って，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，誤差推定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 次を越える初期値を IEEE754 倍精度計算では得ることが出来ないため，山下の方法では 256 次までの分点計算のみ実行してある。また，重みの計算は (9) 式ではなく，後述する Golub & Welsch 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 と Welsch[2] は，1 次から N 次までの (8) を

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

と表現し，これを改めて

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

と置き，分点 $x = \alpha_i$ においては

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

表 1　Legendre, Laguerre, Hermite 直交多項式
Table 1　Legendre, Laguerre and Hermite Orthogonal Polynomials

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

となることを利用し，(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。

現実の分点は全て実数であるが，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

$$J = DTD^{-1}$$

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

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 $\mathbf{q}_i$ の第一成分 $q_1^{(i)}$ 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)4)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，(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極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出来る点にある。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。

誤差推定は，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。丸め誤差の評価及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，もう一度反復を行って停止する。この際，精度の推定を行い，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 次の対称化が行えない。しかし，今回使用した MPFR は指数部が 64bit 環境下では 64bit 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

数値実験を行った環境は以下の通り，x86_64 Dual-core CPU マシンである。従って，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 法及び 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及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，BNCpack のデータ型も全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，内部で実行される全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，(1) 式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。

5.1 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　$\int_0^{\pi/2} \cos x\, dx = 1$

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

表 2　分点の最大値と最小値
Table 2　Maximum and Mimimum values of Abscissas

	Gauss-Legendre		Gauss-Laguerre		Gauss-Hermite	
N	Maximun	Minimum	Maximun	Minimum	Maximun	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

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

丸め誤差の影響を排除するため，この定積分の計算のみ，全て 10 進 5000 桁計算を行っている。この結果を**表 3**，**表 4** に示す。なお，重み w_i は (9) 式で計算できるが，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 & Welsch の方法のどちらで求めた分点でも，重みは (14) に基づいて，全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 3 及び表 4 の結果より，山下の方法，Golub & Welsch の方法，いずれも分点及び重みは全て要求桁数の精度を保っており，過小評価 (下線部分) になった所も，有効桁数一桁以内で収まっている。2 重下線の所は，山下の方法，Golub & Welsch 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。

分点及び重みの大部分は $U+C$ 以上の精度を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。

5.2　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??**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 & Welsch の方法においては一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 1** である。

全体的に Golub & Welsch 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共に上がっており，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，反復回数の減少に繋がっているためである。従って，同じ初期値を QR 法のシフト値として採用できれば，Golub & Welsch の方法は更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推定法に基づいた 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 & Welsch の方法でも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。この結果を踏まえ，本年度中に本プログラムを公開すると共に，今後は古典的誤差推定法の適用範囲の拡大をはかっていきたい。

参　考　文　献

1) Swox AB. GNU MP. http://swox.com/gmp/.
2) G.H.Golub and J.H.Welsch. Calculation of gauss quadrature rules. *Mathematics of Computations*, Vol.23, pp. 221–230, 1969.
3) G.H.Golub and C.F.van Loan. *Matrix Computations (3rd ed.)*. Johns Hopkins University Press, 1996.
4) G.W.Stewart. *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, 1998.
5) G.W.Stewart. *Matrix Algorithms Volume II: Eigen-*

表 3　Gauss 型積分公式の検証 ($\log_{10}$(相対誤差): 山下の方法

Table 3　Verification of Gauss Quadrature Rule ($\log_{10}$(Relative Error)): Yamashita Method

	Gauss-Legendre				Gauss-Laguerre				Gauss-Hermite			
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 4　Gauss 型積分公式の検証 ($\log_{10}$(相対誤差): Golub & Welsch の方法

Table 4　Verification of Gauss Quadrature Rule ($\log_{10}$(Relative Error)): Golub & Welsch method

	Gauss-Legendre				Gauss-Laguerre				Gauss-Hermite			
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

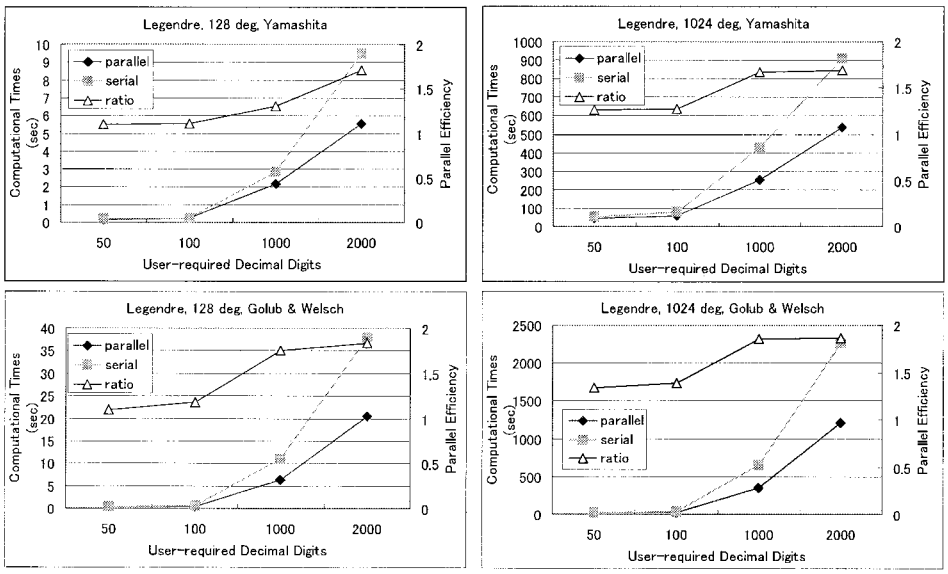

図 1　128(左), 1024 次 (右)Legendre 多項式の分点計算時間と並列化効率: 山下の方法 (上), Golub & Welsch の方法 (下)

Fig. 1　Computational Times and Parallel Efficiencies for 128th(left) and 1024th(right) degree Legendre Orthogonal Polynomials: Yamashita method(uppper), Golub & Welsch method(lower)

systems. SIAM, 2001.

6) 伊理正夫, 藤野和建. 数値計算の常識. 共立出版, 1985.

7) Tomonori Kouya. BNCpack. `http://na-inet.jp/na/bnc/`.

8) MPFR Project. The MPFR library. `http://www.mpfr.org/`.

9) 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.

10) 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.

11) 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.