Vol. 44 No. 10 情報処理学会論文誌 Oct. 2003

二宮法と FLR 法の結合による新しい適応型積分法

日 比 野 勧[†,☆] 長谷川 武光[††] 二 宮 市 三[†††]
細 田 陽 介[††] 佐 藤 義 雄[††]

適応型ニュートン・コーツ則の改良を提案する．積分則で用いる分点数は，積分近似値の精度に関連する．積分区間を分割する方式に基づく適応型積分法では，近似精度と分点数との兼ね合いが重要とされてきた．本論文では，積分則の誤差近似率を考慮したうえで，各部分区間において精度の高い積分則を順次適用するか区間を細分するかの選択基準を組み込むことにより，従来の適応型ニュートン・コーツ積分則を改良している．この改良により，信頼性を失うことなく計算効率が著しく向上した．本適応型積分ルーチンの性能を調べるため，他の著名な積分ルーチンとの性能比較の結果を示す．

A Doubly Adaptive Quadrature Method Based on the Combination of the Ninomiya and the FLR Schemes

Susumu Hibino,[†,☆] Takemitsu Hasegawa,[††] Ichizo Ninomiya,[†††]
Yohsuke Hosoda[††] and Yoshio Sato[††]

An improvement of an adaptive Newton-Cotes quadrature method is proposed. Combining an adaptive Newton-Cotes rule and an algorithm with increasing degree of precision yields an efficient automatic quadrature scheme for univariate integration. Some numerical examples demonstrate the performance of the present quadrature scheme.

1. はじめに

被積分関数 $f(x)$, 積分区間 $[a,b]$, 要求精度 ε が与えられたとき，

$$\left|\int_a^b f(x)dx - S(f)\right| < \varepsilon,$$

を満足する近似値 $S(f)$ を計算する算法を自動積分法[1)]という．この近似値 $S(f)$ は一般に，

$$S(f) = \sum_{j=1}^{n} w_j f(x_j),$$

と表される．ここで，x_j，w_j はそれぞれ，関数標本点（以下分点）と重みである．分点数は計算時間に比例するため，その数は自動積分ルーチンの性能を判断する重要な要素となる．

自動積分法は，主に大域型と適応型の 2 つに分類される．大域型は，被積分関数の性質を大局的にとらえ，その局所的な変化とは無関係にあらかじめ定められた方式に従い，分点数を積分区間全体にわたって一様に増大させていく方法である．一方，適応型は被積分関数を局所的にとらえ，その変化の緩急に分点の密度を適応させることによって，計算効率を図るものである．

適応型積分法では，シンプソン則[5)]等のニュートン・コーツ則に基づくものが主流であり，なかでも二宮の方法[7),11)]は，異常点処理や収束判定の緩和といった機能を備えることで，その優位性を保ってきた．しかし，適応型の苦手とする振動型関数に対しては，Favati, Lotti, Romani（FLR）の方法[3)]が，分点数が少なく優れている．これは，積分値を近似するために，精度の良い高次の積分則を順次適用するか区間分割するかの判断をしながら高精度の近似値を求める方法を用いていることによる．Sugiura ら [9)]も参照のこと．

本来適応型積分法では，分割された各小区間に対し 1 つの固定された積分則を用いるので，要求精度が高くなるほど分割の細分化が進んでしまう．FLR 法は，この弱点を克服するとともに分点を区間端付近に密集させることで，振動型関数にも対応することができる．

† 福井大学大学院工学研究科
Graduate School, Fukui University
†† 福井大学工学部情報・メディア工学科
Department of Information Science, Fukui University
††† 名古屋大学名誉教授
Professor Emeritus of Nagoya University
☆ 現在，福井 CSK
Presently with Fukui CSK

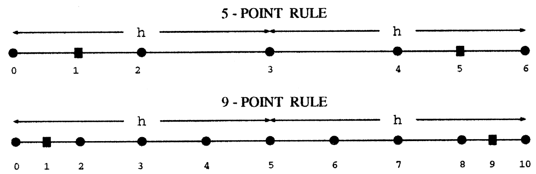

図 1 分点の配置
Fig. 1 Arrangement of sample points.

本論文では，二宮の適応型ニュートン・コーツ積分法に，FLR の方法を組み込んだ複合適応型自動積分ルーチン AQE11D の仕様と算法手順のあらましを説明するとともに，標準テスト関数として有名な Kahaner [6] の 21 問題を用いて，著名な従来のサブルーチンとの性能比較を行った．その結果から，本 AQE11D の性能の良さを示す．

2. 適応型ニュートン・コーツ積分法

ここでは，本論文の方法の基礎となる二宮の適応型ニュートン・コーツ則[11]の概略を述べる．ニュートン・コーツ積分則は，積分区間の分点を等間隔にとった補間型の積分則で，一度計算した関数値を蓄えておいて再利用する適応型積分法には最も適したものである．

二宮は従来の適応型ニュートン・コーツ積分法に，以下の 3 つの改良案を取り入れた．

（1） **誤差推定の強化**：従来の誤差推定法では，ある区間にニュートン・コーツ $2n+1$ 点則を用いて得られる近似値 S と，その区間を等分割しそれぞれの半区間で同じ積分則を施して得られる近似値 S' との差 $|S-S'|$ をとる方法が一般的である．この方法では，$2n+1$ 点則の誤差を推定するために，$2n$ 個の $4n$ 等分点での関数値を新たに計算しなくてはならない．ところで，積分区間の半幅 h に対して，誤差は h^{2n+3} と $2n+2$ 次の微分係数 $f^{(2n+2)}$ に比例している．このため $f^{(2n+2)}$ を直接推定できれば，すなわち $2n+3$ 個の関数値があれば誤差は求まる．ここで，$2n+1$ 点は積分値の導出過程ですでに計算されているため，新たに 2 つの関数値を計算するだけで誤差を推定できることになる．追加される 2 つの関数値は，後述の区間端の異常点処理における有利さを考え，$n=2,4$ の場合の分点の配置で示される**図 1** において，両端近傍の四角で示した 2 点での値である．

（2） **収束判定の緩和**：適応型積分法では，その成り立ちから各々の部分区間において収束判定が行われる．そのときに使用される要求誤差を，単に区間長に比例して定義するのではなく，影響の小さい短い区間に対しては，全積分区間 $[a,b]$ の半幅を $h_0=(b-a)/2$，部分区間の半幅を h_i とおいて，各部分区間の要求精度 ε_i を

$$\varepsilon_i = \varepsilon(h_i/h_0)\sqrt{h_0/h_i}, \quad i=1,2,\cdots \tag{2.1}$$

として積極的に緩和し，収束判定を早い段階で受け入れる．

（3） **異常点の検出・処理**：積分区間が異常点を含む場合，積分近似値が不安定になり，不必要な分割が進む．異常点がどの位置に存在しようと区間を 2^m 等分し続けることで，ある段階で区間端に来る．つまり区間端の異常点に対し，異常点の種類をそれぞれ，不連続点，代数特異点，対数特異点の 3 つに分類し，各々特別な処理を施すことができる[7),11),14)]．しかも，これら異常点の検出には前述の誤差推定での情報を利用しているので新たに関数計算を必要としない．

この 3 つの改良案により，従来の適応型積分法に比べ，関数計算回数が大幅に減少した．しかし，振動型関数への対処，および積分則の精度の低さは否めず，要求精度が高いと関数計算回数が大幅に増える傾向がみられた[12),15)]．

本論文では，二宮の適応型ニュートン・コーツ積分法に FLR[3)]の方法を組み込み，より少ない分点数で計算できるようにする．二宮はニュートン・コーツ 5, 7, 9 点則による適応型積分ルーチンを作成しているが，後述の Recursive Monotone Stable（RMS）積分則との近似精度の兼ね合いから，9 点則を基とする．

3. 積分則列の構成

RMS 積分則[2),4)]は，各部分区間内での大域型積分法であり，そのために前の積分則 $I^{(k-1)}$ の分点を，次の積分則 $I^{(k)}$ で再利用できる．これに関連して，二宮のニュートン・コーツ 9 点則および 11 点則での誤差推定値で使用される分点はすべて再利用できる．この RMS 積分則は，再帰的単調で安定した積分則であり，収束する積分則の列 $I^{(1)},\cdots,I^{(n)}$ において，以下の特徴を持つ．

- $I^{(k)}$ の分点は $I^{(k-1)}$ のすべての分点を含む．
- $I^{(k)}$ の精度は $I^{(k-1)}$ の精度より良い．
- 分点は積分区間の境界付近で間隔が狭くなる．
- 積分則に用いられる重みはすべて数値的に安定している（すなわち正である）．

特に，この積分則列の中で分点数と近似精度との釣合いが最も良い 4 つ[3)]の積分則：13 点則（I_1），19 点則（I_2），27 点則（I_3），41 点則（I_4），を使用する．これに，前章で述べた二宮のニュートン・コーツ 9 点則（安定ではない）に誤差推定のために 2 点を追加した（安

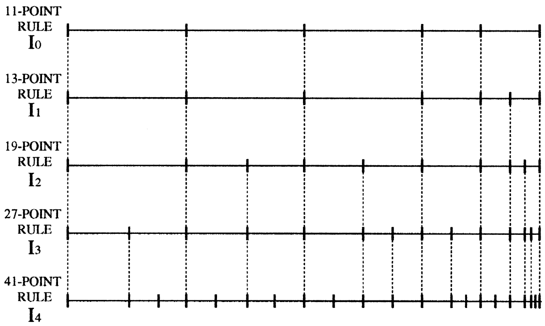

図 2　右半区間における積分則列の分点配置
Fig. 2　Arrangement of sample points in the right-half interval.

定な）11 点則を I_0 とし，積分則列 $\{I_0, I_1, I_2, I_3, I_4\}$ を構成する．

右半区間の分点配置を示す**図 2** から，RMS 積分則列では前の積分則 I_k の分点すべてを次の積分則 I_{k+1} で用いる．その分点配置は，区間を分割した際にも再利用でき，区間の両端近傍で間隔を狭くすることで，振動型関数の局所に対し，早い段階でとらえることができる工夫がなされている．さらに，右半区間において，分割された右区間の分点配置は，その 1 つ前の積分則の配置を 2 分の 1 に納めることになり，分割後は右区間については分点を追加しなくてよい．なお，左半区間については中点に関して対称である．

4. 複合適応型積分法

（1）　選択基準

複合適応型積分法のアルゴリズムにおいて，どの段階で区間分割するか積分則を高次化するべきかを判断する基準が選択基準である．選択基準は，高次の積分則への移行を担う重要な役割を持つため，その過程は慎重に行うべきものである．本論文では，新たに次のような選択基準[10]を組み込んでいる．

ここで，積分則列 $\{Q_1(f), Q_2(f), \cdots, Q_n(f)\}$ が，精度順に並んでいるものする．すなわち $Q_n(f)$ の精度が最も良いとする．被積分関数が積分区間において滑らかで，特に特異性がない場合，$Q_{k-1}(f)$ $(k = 2, 3, \cdots, n)$ からより正確な積分値を算出するため次の $Q_k(f)$ へと移行する．与えられた部分区間における誤差評価は，$Q_{k-1}(f)$ と $Q_k(f)$ を用いれば，

$$E_k(f) = |Q_k(f) - Q_{k-1}(f)|,$$

となり，その値は 0 に収束する：$\lim_{k\to\infty} E_k(f) = 0$．ここで $E_k(f)$ と $E_{k-1}(f)$ を用いて，hint を

$$\mathtt{hint} = E_k(f)/E_{k-1}(f), \tag{4.1}$$

で定義すると，hint ≤ 1 は自明であり，その値は $Q_k(f)$ の値に左右される．つまり $E_k(f)$ が $E_{k-1}(f)$ より小さくなればなるほど，hint の値は小さなものとなる．

hint の値がある与えられたパラメータ P より小さいとき：

$$\mathtt{hint} < P, \tag{4.2}$$

これは被積分関数をとらえた合図となる[10]．すなわち hint の値は，区間上で高次の積分則が適用されるべきタイミングを指摘する．つまり選択基準はパラメータ P に依存するといえる．

以上のことを考慮すると，本論文では，ニュートン・コーツ 9 点則を基としているため，その誤差推定値を e_9 とし，さらに，新たに分点を追加することなく同じ区間における 5 点則の誤差推定値 e_5 も計算できる（図 1 参照）．よって，式 (4.1) にこれらを置き換えると，

$$e_9/e_5 = \mathtt{hint}, \tag{4.3}$$

となる．パラメータ P は，積分則の近似精度によって変化するものであり，その値は固有のものである．本論文では，様々な被積分関数を対象に，P を 0.01 から 0.5 の間で変化させて実際に数値実験を行った．その結果から，積分値を近似するために必要とする関数計算回数が最小となり，特に否定的な要素もなかった $P = 0.2$ に決定する．すなわち，式 (4.3) の hint 値が 0.2 以下なら次数上げ（RMS 則の 13 点則 I_1 を適用）する．そうでないなら現区間を分割する．

（2）　FLR 法

前章で述べた二宮法での 9 点則と 11 点則 (I_0) を計算し，誤差 e_5，e_9 を求め，式 (4.3) と選択基準 (4.2) から次数上げ（高次の積分則への移行）と決定したら，次は RMS 則の 13 点則 I_1 を計算する．絶対誤差 ABSERR$= |I_k - I_{k-1}|$ $(k = 1)$ を求める．一定の条件を満たす場合は ABSERR を小さくすることで[3]，後の収束判定および選択基準を緩和する．次に I_k で使用される関数値の絶対値をとった積分値，

$$\mathrm{RESABS} = \int_a^b |f(x)|dx,$$

を用いた相対誤差基準が，マシン精度 EPMACH に対して十分小さくなった場合，すなわち，

$$\mathrm{ABSERR/RESABS} < 1000 * \mathrm{EPMACH},$$

を満足する場合は，$|I_k| \neq 0.0$ かつ $|I_k| \geq 1.0$ という条件の下で収束判定を行う．ここでは二宮法での緩和法 (2.1) を使用しない．すなわち，

$$\mathrm{ABSERR} \leq |I_k|\,\varepsilon(h_i/h_0), \tag{4.4}$$

を満足する場合は積分近似値を I_k として返し，現区間を終了する．積分近似値 I_k がそれほど小さくなく

十分信頼できるものであり，

$$\mathrm{ABSERR} \le \varepsilon(h_i/h_0), \qquad (4.5)$$

を満足する場合も，同様に I_k を返して終了する．

以上を満足しない場合は，前回の誤差情報を ABSOLD と置くことで，式 (4.1) に ABSERR，ABSOLD を代入した選択基準，

$$\mathrm{ABSERR/ABSOLD} = \texttt{hint}, \qquad (4.6)$$

によって次数上げか区間分割かの判定を行う．ここでの判定基準 (4.2) における P 値は，FLR 法[3)]における固有のパラメータ $P = 0.16$ である．次数上げの場合は，ABSOLD に ABSERR を代入して，次の積分則 I_{k+1} に対し同様の処理を繰り返す．なお，I_4 まで計算された場合は区間を強制的に分割する．

5. 複合適応型積分ルーチン AQE11D

ニュートン・コーツ 9 点則を基にした二宮の 11 点則に FLR 法を組み込んだ本複合適応型自動積分ルーチン AQE11D の仕様とあらましを説明する．

5.1 機　　能

被積分関数 $f(x)$，積分区間 $[a, b]$，許容誤差基準 ε，マシン精度 EPMACH が与えられたとき，

$$\left|\int_a^b f(x)dx - S\right| \le \max(\varepsilon, \mathrm{EPMACH})$$

を満足する近似値 S を計算する．

5.2 使　用　法

AQE11D(a, b, tol, &result, &ncall);

`a`　　　：入力．積分区間の下限．
`b`　　　：入力．積分区間の上限．
`tol`　　：入力．許容絶対誤差 ε．
`&result`：出力．積分の近似値へのポインタ．
`&ncall`：出力．関数計算回数へのポインタ．

プログラムは，C 言語の特性を考慮し，できるだけ視覚的にも分かりやすく作られている．main 関数による AQE11D の呼び出し，被積分関数を登録する function 関数，および各定数を定義した data ファイルを取り込むことで計算される．

5.3 アルゴリズム

手順 1 ··· 初期設定

半幅 h，積分の近似値 result の初期値をそれぞれ $h = h_0 = (b-a)/2$，$\mathrm{result} = 0.0$，その他の諸変数の初期値 (OLD$=0$) と，判定定数各種の値を定める．積分区間 $[a, b]$ の，両端を含む 8 等分点と，最も端に近い 2 つの 16 等分点を加えた，計 11 個の関数値を計算する（図 1 下参照）．手順 2 へ進む．

手順 2 ··· 収束判定

現在の部分区間情報（左端 A，右端 B）から半幅 $h(=h_i)$ を計算して，式 (2.1) の緩和定数 $\sqrt{h_0/h}$ を求める．今現在の積分則の情報を NOW=0 とする．なお，この数値は，図 2 の積分則列の添字に対応する．

下の手順 8 からきた場合，前回（分割前の区間で）使用した積分則の情報 OLD$(=0, 1, \cdots, 4)$（すなわち $I_0, \ldots, I_4$ の添字）から，前回の積分則で用いられた関数値のうち再利用できる点の値を取り出し，9 点積分則および誤差推定 e_9 に必要な 11 点を用意する．その際，計算されていない関数値があれば計算する．e_9 を計算し，現部分区間における要求精度 ε_i を式 (2.1) により求めて収束判定を行う．収束判定が合格ならば手順 6 へ，不合格ならば手順 3 へ進む．

手順 3 ··· 異常点の検査

半幅 h が，h_0，ε から定められる許容最大値 $(\varepsilon^{0.4}h_0)$ よりも大きければ，手順 8 へ行く．もし許容最小値 $(\varepsilon\, h_0)$ よりも小さければ，手順 4 へ進む（詳細は文献 11) を参照）．

それ以外で，式 (4.3) の `hint` が $\texttt{hint} < 0.2$ の条件を満たす場合は，現在の積分近似値を RESULT1$(=I_0)$，NOW=1 とし，手順 7 へ行く． $\texttt{hint} \ge 0.2$ の場合は，現在の部分区間の端に何らかの異常点が存在するかどうかを調べ，存在が認められれば手順 5 へ，そうでないなら手順 8 へ進む．

手順 4 ··· 異常点の再検査

検出規準を弱めて異常点の存在の再確認を行う．合格であれば，手順 5 へ進む．不合格ならばこれ以上の処理を断念し，収束判定不成立のまま現在の区間の近似値を受け入れて，result の値に積算し終了する．

手順 5 ··· 異常点の処理

検出された異常点の種別と特異値によって決められる解析的な計算によって，現在の区間の積分値を定め，手順 6 へ進む．

手順 6 ··· 積分値の計算

現在の区間の近似値，すなわち 9 点則による近似値から推定誤差 e_9 を差し引いた値（11 点則），または異常点に対する特別の処置による近似値を，result の値に積算し終了する．

手順 7 ··· FLR のアルゴリズム

NOW(=1〜4)<OLD ならば，前回の積分則で使用された関数値を取り出し，NOW に対応する積分則に必要な関数値を用意する．NOW≥OLD なら再利用できる分点がないので必要な関数値を計算する．以上の値より，積分値 RESULT2$(=I_1〜I_4)$ 等の計算をする．計算された各値から式 (4.4)，(4.5) により収束判定を行い，合格なら RESULT2 を result に積算し終

表 1 Kahaner のテスト問題集
Table 1 Kahaner's test problems.

NO	A	B	EXACT	INTEGRAND
(1)	0.0	1.0	1.7182818285e+00	exp(x)
(2)	0.0	1.0	7.0000000000e−01	(int) min(x/0.3, 1)
(3)	0.0	1.0	6.6666666666e−01	sqrt(x)
(4)	−1.0	1.0	4.7942822669e−01	0.92*cosh(x)−cos(x)
(5)	−1.0	1.0	1.5822329637e+00	1/(pow(x, 4)+pow(x, 2)+0.9)
(6)	0.0	1.0	4.0000000000e−01	x*sqrt(x)
(7)	0.0	1.0	2.0000000000e+00	1/sqrt(x)
(8)	0.0	1.0	8.6697298734e−01	1/(pow(x, 4)+1)
(9)	0.0	1.0	1.1547006690e+00	2/(2+sin(31.4159*x)
(10)	0.0	1.0	6.9314718056e−01	1/(1+x)
(11)	0.0	1.0	3.7988549304e−01	1/(exp(x)+1)
(12)	0.0	1.0	7.7750463411e−01	x/(exp(x)−1)
(13)	0.1	1.0	9.0986452566e−03	sin(314.159*x)/3.14159*x)
(14)	0.0	10.0	5.0000021117e−01	sqrt(50)*exp(−50*3.14159*pow(x, 2))
(15)	0.0	10.0	1.0000000000e+00	25*exp(−25*x)
(16)	0.0	10.0	4.9936380287e−01	50/3.14159/(2500*pow(x, 2)+1)
(17)	0.01	1.0	1.1213956963e−01	(sin(50.0*3.14159*x)/pow((50.0*3.14159*x)), 2)*50
(18)	0.0	π	8.3867632338e−01	cos(cos(x)+3*sin(x)+2*cos(2*x)+3*sin(2*x)+3*cos(3*x))
(19)	0.0	1.0	−1.000000000e+00	log(x)
(20)	−1.0	1.0	1.5643964441e+00	1/(pow(x, 2)+1.005)
(21)	0.0	1.0	2.1080273550e−01	1/cosh(pow(10*(x−0.2), 2))+1.0/cosh(pow(100*(x−0.4), 4)) +1/cosh(pow(1000*(x−0.6), 6)

了する．

不合格のとき，NOW=4（I_4 まで次数上げが完了している），または式 (4.6) の `hint` が 0.16 以上であれば手順 8 へ行く．そうでなければ NOW に 1 を加え，RESULT1 に RESULT2 を代入して，手順 7 を繰り返す．

手順 8 ··· 区間を分割

現在の区間情報から中点 m を決定し，[A,m]，[m,B] と分け，それぞれを手順 2 へ．付加情報として，現区間の位置，OLD 値に NOW 値を代入して，現在の誤差値，相対誤差等の情報を保持する．

6. 数　値　例

AQE11D と既存の著名なルーチンとの性能を比較するため，積分ルーチンの標準テスト関数として定評のある，Kahaner の 21 問題[6]について数値実験を行った．使用した計算機は，CPU が Pentium4 2.0AGHz の PC（コンパイラオプションなし）で，C 言語を使用した．

Kahaner の問題集は**表 1** にまとめられている．表の各行には，問題番号，積分区間の下限 A，上限 B，積分真値 EXACT，および被積分関数を順に示した．積分真値としては，10 進相当 14 桁の精度で行われた各ルーチンの実行結果の平均値を使用している．

比較対象となるサブルーチンは，二宮の適応型ニュートン・コーツ積分法の 9 点則による AQNN9[11]，Favati らの DQXG[3]において，4 つの RMS 積分則を用いる KEY=2 の DQXG2 である．

性能比較の結果が，許容絶対誤差の 3 つの値 10^{-3}，10^{-6}，10^{-9} に対して，それぞれ**表 2**，**表 3**，**表 4** に収められている．これらの表では，ERROR の見出しの下に積分真値との絶対誤差を記し，その数値の右につけられた星印は，誤差が基準を満足していないことを意味している．N の見出しの下に関数計算回数，さらに最下行には，21 問全体に対する成功率と平均関数計算回数が付け加えられている．そして，CPU TIME の見出しの下に平均実行時間が与えられている．これは，実行プログラムの CPU 占有率が 9 割以上確保できたことを前提に，各ルーチンを 100 万回繰り返し実行した平均時間で，単位はマイクロ秒（10^{-6} 秒）である．

なお，本 AQE11D が AQNN9 を基にしていることや，FLR 法を組み込んだときの負荷を調べるため，二宮の AQNN9 と本 AQE11D の厳密な実行時間を計る必要がある．そこで，AQNN9 として，Net-NUMPAC[17]で提供されている NUMPAC サブルーチンの AQNN9D のソースプログラム（Fortran77 言語で書かれた）を，C 言語に変換したものを使用した．二宮の論文[11]での AQNN9D の数値実験結果（IBM16 進計算機を使用）より上記の方法での AQNN9 の結

表 2 適応型積分ルーチンの性能比較
Table 2 Comparison of performance of adaptive quadrature routines.

ERROR REQUIREMENT 1.0e−03									
	DQXG2			AQNN9			**AQE11D**		
NO	N	ERROR	CPU TIME	N	ERROR	CPU TIME	N	ERROR	CPU TIME
(1)	19	2.2e−16	23.4	11	2.2e−16	3.58	11	2.2e−16	3.90
(2)	357	1.8e−05	359.	111	2.9e−05	11.3	111	2.9e−05	15.1
(3)	107	5.7e−06	97.9	21	4.0e−04	3.50	21	4.0e−04	4.28
(4)	41	2.2e−16	40.1	11	3.8e−14	5.79	11	3.8e−14	5.71
(5)	41	1.3e−15	24.5	21	2.7e−08	2.61	21	2.7e−08	3.36
(6)	27	4.4e−08	21.0	11	9.2e−06	2.07	11	9.2e−06	2.35
(7)	747	2.1e−04	752.	81	1.6e−10	13.5	41	1.2e−10	8.71
(8)	41	2.2e−16	24.1	11	2.5e−07	1.65	11	2.5e−07	1.96
(9)	241	1.2e−11	224.	81	1.0e−05	17.1	83	1.0e−05	23.5
(10)	41	2.2e−16	23.3	11	3.9e−10	1.62	11	3.9e−10	1.92
(11)	27	2.2e−16	28.5	11	2.4e−14	3.89	11	2.5e−14	4.25
(12)	27	2.2e−16	24.7	21	1.1e−16	6.83	21	1.1e−16	7.54
(13)	641	2.2e−16	624.	321	6.2e−07	71.7	323	6.2e−07	86.0
(14)	147	9.2e−09	199.	71	8.1e−10	37.5	67	8.7e−09	41.5
(15)	107	1.5e−12	105.	61	1.1e−06	17.9	57	9.8e−11	21.3
(16)	201	1.9e−06	169.	91	5.2e−07	11.6	93	5.2e−07	17.1
(17)	587	4.8e−09	581.	101	7.4e−04	25.5	101	7.4e−04	32.5
(18)	81	1.6e−12	154.	51	5.4e−06	60.4	49	2.4e−06	42.8
(19)	385	4.6e−05	457.	81	6.0e−10	19.1	31	3.4e−11	9.23
(20)	41	0.0e+00	24.0	11	3.1e−05	1.82	11	3.1e−05	2.18
(21)	241	1.1e−03*	44.6	61	1.1e−03*	69.2	61	1.1e−03*	66.8
	197	95%		60	95%		55	95%	

表 3 適応型積分ルーチンの性能比較
Table 3 Comparison of performance of adaptive quadrature routines.

ERROR REQUIREMENT 1.0e−06									
	DQXG2			AQNN9			**AQE11D**		
NO	N	ERROR	CPU TIME	N	ERROR	CPU TIME	N	ERROR	CPU TIME
(1)	19	2.2e−16	23.3	11	2.2e−16	3.60	11	2.2e−16	3.89
(2)	673	1.7e−08	663.	211	2.9e−08	21.2	211	2.9e−08	28.7
(3)	387	4.0e−09	379.	91	2.5e−12	14.0	41	2.4e−12	8.81
(4)	41	2.2e−16	40.0	11	3.8e−14	5.78	11	3.8e−14	5.68
(5)	41	1.3e−15	24.6	41	1.0e−11	4.51	25	5.0e−10	5.68
(6)	67	7.8e−09	55.7	41	5.1e−08	6.54	19	2.6e−07	5.31
(7)	1547	2.0e−07	1530.	91	1.7e−10	14.9	41	1.2e−10	8.67
(8)	41	2.2e−16	24.2	21	3.3e−11	2.66	13	1.2e−08	3.16
(9)	321	4.0e−14	311.	221	9.3e−09	46.1	195	3.6e−11	66.2
(10)	41	2.2e−16	23.3	11	3.9e−10	1.68	11	3.9e−10	1.90
(11)	27	2.2e−16	28.5	11	2.4e−14	3.87	11	2.5e−14	4.23
(12)	27	2.2e−16	24.8	21	1.1e−16	6.81	21	1.1e−16	7.54
(13)	641	2.2e−16	624.	641	1.1e−10	142.	531	2.5e−09	180.
(14)	173	3.5e−14	241.	91	3.4e−10	44.6	81	3.5e−14	47.4
(15)	147	2.2e−16	149.	81	5.5e−10	23.9	71	8.9e−16	26.2
(16)	281	4.6e−12	249.	101	1.5e−09	12.7	99	3.6e−10	20.0
(17)	641	3.8e−13	621.	481	5.6e−09	118.	381	5.8e−09	132.
(18)	81	1.6e−12	153.	111	1.6e−09	131.	85	7.3e−11	74.8
(19)	757	2.9e−08	881.	91	2.8e−10	21.2	31	3.4e−11	9.23
(20)	41	0.0e+00	24.0	21	1.1e−08	3.02	27	2.0e−12	6.51
(21)	241	1.1e−03*	44.6	111	1.1e−03*	127.	105	1.1e−03*	119.
	297	95%		120	95%		96	95%	

果が概して良好（少ない標本点数）である理由は，本実験が IEEE 浮動小数方式[8)]の計算機を使用していることによるものと想像される．

DQXG2 としては，Fortran77 言語で作成されているソースプログラム[16)]をオブジェクト化したものを，本方法のメイン関数にインポートする形をとった．よっ

表 4　適応型積分ルーチンの性能比較
Table 4　Comparison of performance of adaptive quadrature routines.

ERROR REQUIREMENT 1.0e−09									
	DQXG2			AQNN9			**AQE11D**		
NO	N	ERROR	CPU TIME	N	ERROR	CPU TIME	N	ERROR	CPU TIME
(1)	19	2.2e−16	23.4	11	2.2e−16	3.59	11	2.2e−16	3.90
(2)	1017	1.7e−11	955.	311	2.8e−11	30.7	311	2.8e−11	41.6
(3)	667	2.7e−12	612.	131	2.5e−12	19.8	41	2.4e−12	8.81
(4)	41	2.2e−16	40.0	21	5.6e−17	11.4	13	8.3e−16	7.50
(5)	41	1.3e−15	24.6	61	1.8e−11	6.64	37	1.1e−13	8.54
(6)	227	7.6e−12	215.	91	8.9e−12	13.8	69	2.4e−10	25.5
(7)	2347	2.0e−10	2240.	251	1.7e−13	39.5	47	1.0e−12	12.4
(8)	41	2.2e−16	24.0	41	5.0e−15	4.72	27	3.3e−16	6.25
(9)	321	4.0e−14	311.	421	5.8e−12	87.7	267	2.5e−13	86.7
(10)	41	2.2e−16	23.4	21	3.9e−13	2.61	13	3.8e−12	3.12
(11)	27	2.2e−16	28.5	11	2.4e−14	3.88	11	2.5e−14	4.23
(12)	27	2.2e−16	24.7	21	1.1e−16	6.84	21	1.1e−16	7.53
(13)	641	2.2e−16	624.	1271	6.3e−14	281.	763	7.7e−15	255.
(14)	213	0.0e+00	289.	131	2.8e−12	59.1	93	1.4e−14	59.6
(15)	147	2.2e−16	149.	121	5.5e−13	35.3	71	8.9e−16	26.2
(16)	321	2.2e−16	294.	201	4.8e−12	24.6	124	1.6e−12	29.8
(17)	641	3.8e−13	621.	981	1.1e−12	239.	615	1.5e−12	227.
(18)	121	1.9e−14	238.	201	1.1e−12	237.	93	1.4e−14	81.8
(19)	1101	1.3e−11	1250.	171	1.7e−11	39.3	35	3.0e−13	12.3
(20)	41	0.0e+00	24.1	61	2.3e−13	7.59	41	0.0e+00	9.06
(21)	241	1.1e−03*	44.6	221	1.1e−03*	252.	147	1.1e−03*	170.
	394	95%		226	95%		136	95%	

て，DQXG2 の実行時間については厳密な値ではなく，あくまで参考程度であることに注意されたい．

ここで，これらの表から次のような事実が認められるであろう．

（1）　比較的穏やかな関数の問題（4，8，10，11）では，二宮の 11 点則から RMS の 13 点則への移行が非常に効率良く行えている．選択基準とそのパラメータ値における次数上げ判定が，うまく機能していると判断できる．さらに，RMS 積分則の精度の高さもうかがえるであろう．

（2）　適応型積分法が苦手とする振動型の問題（9，13，17，18）に対しては，本方法では AQNN9 より関数計算回数が少ないが，要求精度 10^{-9}（表 4）に対して DQXG2 より多い．これは，選択基準の違いからくるものと推測する．

（3）　異常点を持つ問題（2，3，7，19）は，DQXG2 が苦手とする．しかし，本 AQE11D は異常点処理によってこの弱点を補っている．特に問題 (3)，(7)，(19) に対しては，異常点処理と FLR 法の連係によって，本方法での関数計算回数が大幅に減少した．

（4）　ピーク型の問題（14，15，16，21）については，適応型積分法が最も得意とするところである．ここでも，本 AQE11D は他のルーチンより関数計算回数が少なく，優れているといえる．しかし，問題 (21) のような小区間におけるピーク関数には，基本的にどのルーチンも要求精度を満たせなかった．これは $x = 0.6$ にある最も鋭いピークが，分点の網から完全に洩れてしまうためである．

（5）　実行平均時間に関して，FLR 法に大きな時間が費されていることが分かった．FLR 法内における収束判定を行うための計算量が大きいためと思われる．二宮の AQNN9 は，積分則を 1 つしか使わないためアルゴリズムが複雑にならず，非常に効率良く計算しているといえる．そのため，本 AQE11D では，関数計算回数が AQNN9 より少ないにもかかわらず平均実行時間が遅いことが判明した．なお，関数値計算のオーバヘッドにもよるが，AQE11D が AQNN9 を速度的に上回るには，関数計算回数の比がおよそ 1.8 倍以上なければならない．それらをふまえた結果，要求精度が低いときは AQNN9，高いときは AQE11D と二極化してしまった．

以上の考察から，少なくとも Kahaner の問題集に関して，そしておそらくは実際の計算に現れる多くの問題において，現在ある可能な自動数値積分ルーチンの中で，本 AQE11D は非常に優秀な性能を持つといえる．

7. あとがき

従来の適応型積分ルーチンに，FLR 法を組み込むことで，性能を大幅に増進させることに成功した．今後の課題としては，誤差近似率の加速性を考慮したパラメータ値の動的な評価，そして実行処理時間の短縮策として，FLR 法の簡略化がある．

参考文献

1) Davis, P. J. and Rabinowitz, P.: *Methods of Numerical Integration*, 2nd edition, Academic Press (1984).
2) Favati, P., Lotti, G. and Romani, F.: Interpolatory integration formulas for optimal composition, *ACM Trans. Math. Softw.*, Vol.17, No.2, pp.207–217 (1991).
3) Favati, P., Lotti, G. and Romani, F.: ALGORITHM 691 Improving QUADPACK automatic integration routines, *ACM Trans. Math. Softw.*, Vol.17, No.2, pp.218–232 (1991).
4) Favati, P., Fiorentino G., Lotti, G. and Romani, F.: Local error estimates and regularity tests for the implementation of double adaptive quadrature, *ACM Trans. Math. Softw.*, Vol.23, No.1, pp.16–31 (1997).
5) Gander, W. and Gautschi, W.: Adaptive quadrature — revisited, *BIT*, Vol.40, pp.84–101 (2000).
6) Kahaner, D.K.: Comparison of numerical quadrature formulas, *Mathematical Software*, Rice, J.R. (Ed.), Academic Press, pp.229–259 (1971).
7) Ninomiya, I.: Improvements of adaptive Newton-Cotes quadrature methods, *J. Information Processing*, Vol.3, No.3, pp.162–170 (1980).
8) Overton, M.L.: *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia (2001).
9) Sugiura, H. and Sakurai, T.: On the construction of high-order integration formulae for the adaptive quadrature method, *J. Comput. Appli. Math.*, Vol.28, No.1–3, pp.367–381 (1989).
10) Venter, A. and Laurie, D.P.: A doubly adaptive integration algorithm using stratified rule, *BIT*, Vol.42, No.1, pp.183–193 (2002).
11) 二宮市三：適応型ニュートン・コーツ積分法の改良，情報処理学会論文誌，Vol.21, No.5, pp.504–513 (1980).
12) 日比野勧：適応型数値積分ルーチンの性能比較，福井大学工学部平成十二年度卒業論文．
13) 日比野勧：適応型ニュートン・コーツ積分則の次数上げアルゴリズムを用いた改良，福井大学大学院工学研究科平成十四年度修士論文．
14) 富士通：FACOM FORTRAN SSL II 使用手引書，pp.110–113 (1980).
15) 芳沢和紀：適応型数値積分ルーチンの性能比較，福井大学工学部平成十三年度卒業論文．
16) http://www.netlib.org/toms/691/
17) http://netnumpac.fuis.fukui-u.ac.jp/

(平成 15 年 4 月 21 日受付)
(平成 15 年 9 月 5 日採録)

日比野　勧

昭和 53 年生．平成 13 年福井大学工学部情報工学科卒業．平成 15 年同大学大学院工学研究科博士前期課程修了．現在，株式会社福井 CSK に在職．

長谷川武光（正会員）

昭和 19 年生．昭和 47 年名古屋大学大学院工学研究科博士課程単位修得退学．工学博士．福井大学工学部情報・メディア工学科教授．主たる研究テーマは数値解析，数学ソフトウェアおよびインターネット上での数値計算環境の構築．日本応用数理学会，日本物理学会，AMS，IEEE（CS）各会員．

二宮　市三（正会員）

大正 10 年生．昭和 18 年 9 月東京帝国大学工学部航空学科機体専修卒業．昭和 20 年より 40 年間名古屋大学工学部に奉職，同大学大学院工学研究科情報工学専攻教授を経て昭和 60 年定年退職．引き続き中部大学経営情報学科教授として 9 年間勤務．計算機用数値計算法の研究に従事．汎用数値計算ライブラリ NUMPAC の構築と普及に尽力．昭和 55 年情報処理学会創立 25 周年記念論文賞受賞．日本応用数理学会会員．

細田　陽介（正会員）

昭和 40 年生．平成 6 年名古屋大学大学院博士課程修了．広島市立大学情報科学部助手，富山県立大学工学部助手を経て，現在福井大学工学部情報・メディア工学科助教授．博士（工学）．数値解析，特に悪条件な線形方程式の数値解法の研究に従事．日本応用数理学会会員．

佐藤　義雄（正会員）

昭和 23 年生．昭和 46 年名古屋大学工学部応用物理学科卒業．昭和 50 年同大学大学院工学研究科情報工学専攻修士課程修了．昭和 53 年同博士課程修了．現在，福井大学工学部情報・メディア工学科助手．教育・研究支援用グラフィカルユーザインタフェース，数学ソフトウェアの改良・開発等の研究に従事．