JOURNAL OF COMPLEXITY 10, 296–340 (1994)

Asymptotic Behavior of Automatic Quadrature*

PAOLA FAVATI

IEI-CNR Via S. Maria 46, 56100 Pisa, Italy

GRAZIA LOTTI

Dipartimento di Matematica, University of Parma, Via M. D'Azeglio 85/A, 43100 Parma, Italy

GASPARE DI MARCO AND FRANCESCO ROMANI†

Dipartimento di Informatica, University of Pisa, Corso Italia 40, 56100 Pisa, Italy

Received May 13, 1992

The computational cost of automatic quadrature programs is analyzed under the hypothesis of exactness (or asymptotic consistence) of local error estimates. The complexity measure used, in this work, is the number N of function evaluations in real exact arithmetic seen as a function of the number E of exact decimal digits in the result. The methods of integration reviewed are m-panel rules, Clenshaw–Curtis quadrature, global adaptive quadrature, double exponential quadrature. For m-panel and global adaptive quadrature, based on a local rule of degree $r - 1$ the constants hidden by the "O" notation are determined in terms of the derivatives of the integrand and of the numerical properties of the local rule. Two new algorithms are introduced, called double-adaptive quadrature and triple-adaptive quadrature, which achieve outstanding performances on several classes of integrands.

1. INTRODUCTION

We study the asymptotic complexity of computing the exact real representation of the definite integral

* Work supported by CNR Grant 93.00570.CT01.

† To whom correspondence should be addressed.

0885-064X/94 $6.00

$$I = \int_a^b f(x)\,dx, \qquad a < b, \tag{1.1}$$

by using an automatic integration routine.

The complexity measure used in this analysis is the number N of function evaluations versus the number of exact decimal digits in the result. More precisely, once given an algorithm that computes (1.1) with any accuracy, the kth step of the integration process produces a value $I^{(k)}$ that approximates the integral with an absolute error $I - I^{(k)}$. Let $E^{(k)} = -\log_{10}|I - I^{(k)}|$ and $N^{(k)}$ be the overall number of function values computed from step 1 to step k. The pairs $(E^{(k)}, N^{(k)})$, $k = 1, 2, \ldots$, can be viewed as instances of a function $N(E)$, whose asymptotic behaviour is investigated here. From the point of view of the numerical analysis it is more natural to study the behaviour of a quadrature algorithm by measuring $|I - I^{(k)}|$ as a function of the number of evaluations $N^{(k)}$; this last measure can be easily converted to our measure. It is worth noting that a more detailed analysis would be required to estimate the computational load of the algorithm by taking into account the cost of the arithmetic operations, the complexity of the integrand, the computational cost of the error estimation, and the overhead due to the algorithm itself.

Analysis of the asymptotic complexity of numerical quadrature algorithms is a hard task, widely studied in the literature from several points of view (e.g., see Rice (1975), Traub *et al.* (1988)). Part of this work overlaps much of the results present in the literature; nevertheless we feel that a uniform presentation of the topic is fully justified.

The various approaches to this problem differ especially in the classes of integrands considered and in the simplifying assumptions which are adopted for the algorithms under investigation. Our choices are explained in Section 2, where the main notations, the basic assumptions, and the classes of integrands taken into consideration are introduced. In Sections 3 and 4 the complexities of nonadaptive quadrature and of classical global adaptive quadrature are reviewed. We also derive explicit expressions for the constants associated to the asymptotic cost of the panel nonadaptive algorithm and global adaptive algorithm. In Section 5 a double-adaptive quadrature algorithm and a triple-adaptive one, which achieve outstanding performances, are introduced. In Section 6 the complexity bounds are summarized. Some auxiliary results are given in the Appendix.

For any algorithm, plots of the complexity function $N(E)$, obtained by extensive computations carried out with high precision arithmetic, are presented. Moreover, a special type of graphical presentation (see Fig. 2.3 below) is used, allowing a full insight of the evolution of the algorithms.

2. PRELIMINARIES

Notation

Given two real functions $f(x)$, $g(x)$, and a point $x_0 \in \mathbb{R} \cup \{+\infty\}$,

$f(x) = O(g(x))$	means that	$\limsup_{x \to x_0} \lvert f(x)/g(x) \rvert = c,\ c \in \mathbb{R}$;
$f(x) = o(g(x))$	means that	$\lim_{x \to x_0} \lvert f(x)/g(x) \rvert = 0$;
$f(x) = \Omega(g(x))$	means that	$g(x) = O(f(x))$;
$f(x) = \Theta(g(x))$	means that	$f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$;
$f(x) \sim g(x)$	means that	$\lim_{x \to x_0} \lvert f(x)/g(x) \rvert = 1$;
$f(x) \lesssim cg(x)$	means that	$\limsup_{x \to x_0} \lvert f(x)/g(x) \rvert \leq c,\ c \in \mathbb{R}$.

In the following, the point x_0 will be implicitly defined by the context.

Classes of Integrands

A complexity analysis of quadrature algorithms can be made only under precise definitions on the class of the integrand taken into consideration. The first class we consider is that of analytic functions in the integration interval.

Class $\mathcal{A}[a, b, \delta]$ (Analytic). Analytic functions in $[a, b]$, continuable analytically to be single-valued and regular in a region $\Gamma(a, b, \delta)$, $\delta > 0$. $\Gamma(a, b, \delta)$ is defined as the region of the complex plane obtained as the closed union of two semicircles of radius δ and centres at a and b, respectively, and the rectangle of vertices $(a, \pm\delta)$, $(b, \pm\delta)$; $\partial\Gamma$ denotes the boundary of Γ (see Fig. 2.1).

The second class is that of analytic functions in the integration interval, except in a finite number of singular points. Moreover, in order to bound the error produced by a local quadrature rule (Theorems A.2 and A.3 in

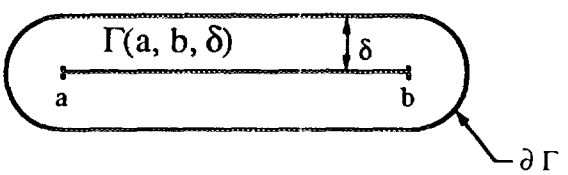

FIG. 2.1. The region $\Gamma(a, b, \delta)$.

the Appendix) we allow the function to be unbounded only at the endpoints of the integration interval.

Class $\mathcal{M}[a, b, s]$ (Multiple singularities). A function f in $\mathcal{M}[a, b, s]$ has the following properties:

(1) there exists a finite number $k \geq 1$ of singular points $a \leq y_1 < y_2 < \cdots < y_k \leq b$.

(2) if $s = -1$, at least one discontinuity in the function is present; if $s \geq 0$, f is in $C^s[a, b]$ with at least one singularity on the $(s + 1)$th derivative; if $s = \infty$, f is in $C^\infty[a, b]$.

(3) f is analytic in $[a, b] - \{y_1, y_2, \ldots, y_k\}$ and bounded in any closed subinterval of (a, b).

(4.a) if $y_1 = a$ then there exist two constants $\delta_2 > \delta_1 > 0$ such that f is continuable analytically in the interior of $\Gamma(a + \delta_1, a + \delta_2, \delta_1)$, and

$$\inf \{\beta : |f(z)| = O(|z - a|^{-\beta}), z \in \Gamma(a + \delta_1, a + \delta_2, \delta_1)\} < 1;$$

(4.b) if some $y_i \in (a, b)$ then there exist two constants $\delta_2 > \delta_1 > 0$ such that f is continuable analytically in the interior of both $\Gamma(y_i - \delta_2, y_i - \delta_1, \delta_1)$ and $\Gamma(y_i + \delta_1, y_i + \delta_2, \delta_1)$, and

$$\inf \{\beta : |f(z)| = O(|z - y_i|^{-\beta}), z \in \Gamma(y_i - \delta_2, y_i - \delta_1, \delta_1)\} < 1,$$
$$\inf \{\beta : |f(z)| = O(|z - y_i|^{-\beta}), z \in \Gamma(y_i + \delta_1, y_i + \delta_2, \delta_1)\} < 1;$$

(4.c) if $y_k = b$ then there exist two constants $\delta_2 > \delta_1 > 0$ such that f is continuable analytically in the interior of $\Gamma(b - \delta_2, b - \delta_1, \delta_1)$ and

$$\inf \{\beta : |f(z)| = O(|z - b|^{-\beta}), z \in \Gamma(b - \delta_2, b - \delta_1, \delta_1)\} < 1.$$

Properties 3 and 4 ensure that f is integrable in $[a, b]$ and that the local quadrature error can be bounded.

Sometimes, for functions of class $\mathcal{M}$, an additional property can be useful, which ensures that the rth derivative of f has a definite behaviour near any singularity point:

$$\begin{aligned} |f^{(r)}(x)| &= \Omega((x - y)^{-\alpha_+(y)+\varepsilon-r}), \quad \text{for any } \varepsilon > 0,\ x \to y,\ x > y, \\ |f^{(r)}(x)| &= \Omega((y - x)^{-\alpha_-(y)+\varepsilon-r}), \quad \text{for any } \varepsilon > 0,\ x \to y,\ x < y, \end{aligned} \tag{2.1}$$

where

$$\alpha_+(y) = \inf \{\beta : |f^{(r)}(x)| = O((x - y)^{-\beta-r}), x > y\};$$
$$\alpha_-(y) = \inf \{\beta : |f^{(r)}(x)| = O((y - x)^{-\beta-r}), x < y\}.$$

Two interesting special cases of class $\mathcal{M}$ are the following:

Class $\mathcal{PP}$ (Piecewise polynomials). These are functions of the type

$$f(x) = \begin{cases} p(x), & x \le y, \\ q(x), & x > y, \end{cases}$$

to be integrated on $[a, b]$, $a \le y \le b$, where $p(x)$ and $q(x)$ are polynomials. In this case, if the rule used is precise enough, the only source of error is the singularity.

Class $\mathcal{ES}$ (End-point singularities). These are integrable analytic functions in (a, b), with algebraic or logarithmic singularities at the endpoints. Class $\mathcal{ES}$ is essentially the same as that used by Takahasi and Mori (1974).

The Assumptions of Our Ideal Model

In general the analysis of the asymptotic complexity of numerical quadrature cannot be made, without some simplifying assumptions. The critical problems that need to be simplified are the roundoff error management and the truncation error estimates. We make the following two main assumptions.

Assumption 2.1. We assume exact real arithmetic is used, or equivalently, a multiple precision arithmetic with a variable number of digits sufficient to neglect roundoff errors.

Assumption 2.2. For any application of a local quadrature rule let e_{est} be the estimate of the error used in the algorithm and let e_{true} be the true error. We assume that

$$e_{\text{est}} = e_{\text{true}}.$$

In the Appendix, it will be proved that, for regular enough functions, Assumption 2.2 is asymptotically true. Indeed, for interpolatory rules with degree of precision $r - 1$, applied on a subinterval of length 2λ to $C^{r'+2}$, $r' > r$, functions, there exists a simple error estimate such that

$$e_{\text{est}} = e_{\text{true}} + O(\lambda^{r'+1}),$$

where

$$e_{\text{true}} = C\lambda^{r+1} + O(\lambda^{r+3}), \qquad C \text{ a constant independent of } \lambda.$$

This is in some sense an extension of the concept of asymptotic consistence of local error estimates given in Kahaner and Stoer (1983).

These assumptions force us to study ideal programs and the results are not directly applicable to practical environments, but this approach has the following advantages:

(1) The analysis of automatic quadrature algorithms with exact arithmetic and exact error bounds allows investigating the problems related to the integration strategy (choice of the rules, type of algorithm, etc.) without considering the influence of the error estimate techniques.

(2) All the questions concerning failures, successes, abnormal terminations, and the computational load due to error estimation can be avoided.

(3) The performance of an algorithm with an exact error estimate is clearly the best one among those achieved by the same algorithm with any practical error estimating strategy. Therefore, this approach allows an absolute rating of different error estimating strategies.

Number of Evaluations

In the context of automatic quadrature programs, the composition arises often by bisections; a particular sequence of subintervals is obtained by the repeated bisection of intervals. The total number of evaluations depends on the geometrical properties of the rules used. There exist rules that allow reusing the previously computed function values: the most common examples are Newton–Cotes rules, but they can be used only with a low degree of precision. RMS rules (Favati *et al.*, 1991a) have the same geometrical property and can be applied with a large number of points. For our purposes, it is sufficient to observe that in any case the number of function evaluations N is proportional to the number of nodes n of the rule used and the number m of subintervals,

$$N = v_1 m + v_0;$$

e.g., for Newton–Cotes and RMS rules $v_0 = 1$, $v_1 = n - 1$, for Gauss–Legendre and Gauss–Kronrod rules $v_0 = -n$, $v_1 = 2n$.

In common practice the local quadrature is based on a pair of formulas: one is used to estimate the integral and the other one is used to get an estimate of the error. In order to take into consideration this approach in our model, we should measure the cost of applying both formulas.

Algorithm Presentation

The quadrature algorithms are presented in the form of Pascal-like programs; the following procedure heading is used:

```
procedure Quadrature(a, b, epquad: real; var abserr,
result: real),
```

where

on input `a, b` are the extremes of the integration interval;
`epquad` is the absolute error tolerance specified by the user.
on output `result` is the integral value estimated by the program
`abserr` is the estimate of $|I - \texttt{result}|$ returned by the program

The quadrature programs use a local routine

```
procedure rule(a, b: real; k: integer; var abserr, result: real);
```

that computes:

`result` an approximation of the integral obtained by applying the k-th element of a family of rules $\mathcal{Q}_1, \ldots, \mathcal{Q}_k, \ldots$ of increasing accuracy.
`abserr` the absolute value of the exact error.

The algorithms are presented in a very schematic form, without any roundoff checking or exceptional conditions handling, using Assumptions 2.1 and 2.2.

Graphical Presentations of Numerical Experiments

To improve the clearness of the presentation, the results of numerical experiments will be presented by means of two types of graphs. The plot of the complexity function allows comparing the overall behaviour of different algorithms or of the same algorithm on different integrands (Fig. 2.2).

Another type of plot (Fig. 2.3) allows deep insight into the structure of the computation of a program based on the composition of elementary rules. For any subinterval $[x, y]$ let m be the number of evaluations and let e be the true error; then $d = m/(y - x)$ is the density of the function evaluations in that subinterval. The quantities $|e|$ and d are plotted in decimal logarithmic scale versus the integration interval: the ordinates of black points are $-\log_{10}|e|$ and those of grey points are $-\log_{10}d$, while the abscissas are the midpoints of the corresponding subintervals. In addition, the number of decimal digits of accuracy for the whole integration interval, the total number of function evaluations, and the number of subintervals are printed at the top of the graph.

All the computations were carried out with the help of *Mathematica*™ on an Apple® Macintosh™ and a workstation Sun™, using 70 decimal digit arithmetic.

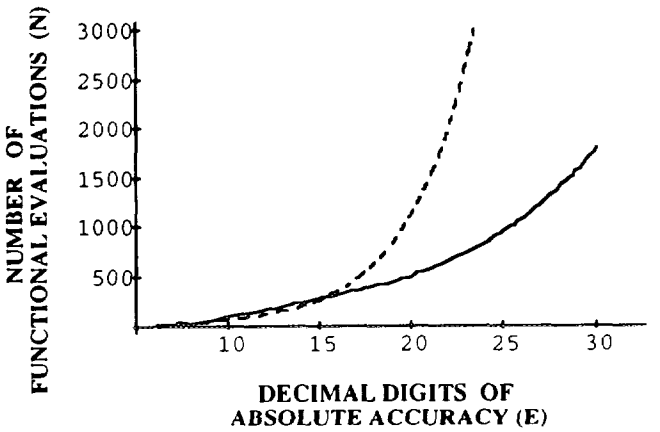

FIG. 2.2. Plot of the complexity function of two quadrature routines.

3. NONADAPTIVE QUADRATURE SCHEMATA

Clenshaw–Curtis Quadrature

Algorithm 3.1 outlines a nonadaptive automatic quadrature routine based on a sequence of different rules of increasing precision. Probably, the best choice is the family of Clenshaw–Curtis formulas (Clenshaw and

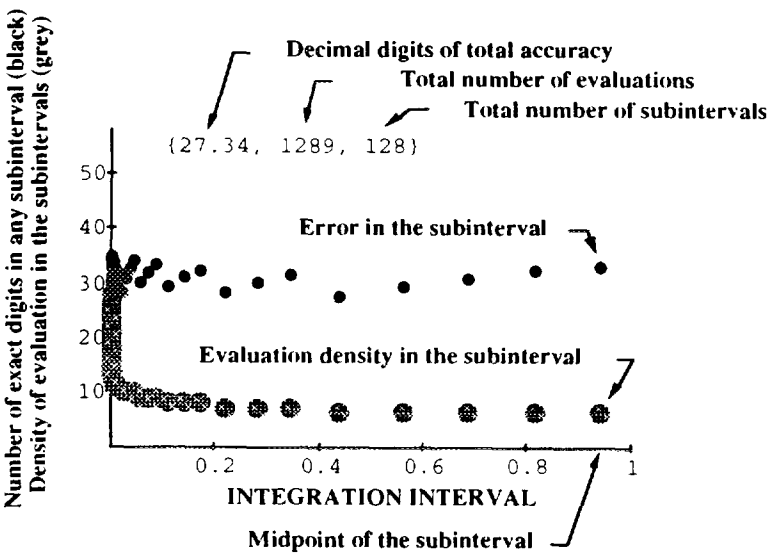

FIG. 2.3. Plot of the status of a quadrature routine.

Curtis, 1960) which can be computed, with any number of nodes, via the fast Fourier transform algorithm without precomputing weights and nodes (Gentleman, 1972).

ALGORITHM 3.1.

```
procedure CCQuadrature(a, b, epquad: real; var abserr, result:
          real);
    var k: integer;
{   procedure CCrule(a,b: real; k: integer; var abserr, result:
              real);     }
{   implements Clenshaw-Curtis Quadrature with 2^k + 1  points      }

begin
    k:=1;
    CCrule(a, b, k, abserr, result);
    while abserr > epquad do begin
         k:=k+1;
         CCrule(a, b, k, abserr, result);
    end;
end;
```

The number of evaluations after k steps is $N = 2^k + 1$. This algorithm converges very well for regular functions.

From the results of Chawla (1968) and Riess and Johnson (1972) it is easy to prove the following theorem.

THEOREM 3.1. *Let f be in the class $\mathcal{A}[a, b, \tau(b - a)]$, $\tau > 0$, and let $I(m)$ be the result of the integration by a Clenshaw–Curtis rule with m nodes. Then there exist two constants F and m_0 for which*

$$|I - I(m)| \leq F \frac{b - a}{2} \max_{z \in \partial\Gamma(a,b,(b-a)\tau)} |f(z)|\, \rho^{-m-2},$$

where

$$m \geq m_0, \qquad \rho = 2\tau + (1 + 4\tau^2)^{1/2} > 1 + 2\tau.$$

Then, for any $f \in \mathcal{A}[a, b, \tau(b - a)]$, $\tau > 0$, we get the bound

$$|I - I(N)| \leq C\rho^{-(N+2)}, \qquad \rho > 1 + 2\tau, \tag{3.1}$$

where C is independent of N and

$$E = -\log_{10}|I - I(N)| \geq (N + 2)\log_{10}\rho - \log_{10}C,$$

whence

$$N = O(E). \tag{3.2}$$

Nonadaptive Composition of a Fixed Rule

Algorithm 3.2 outlines a generic nonadaptive automatic quadrature routine based on panel rules. This schema is called PNAQ in the following.

ALGORITHM 3.2.

```
procedure PNAQuadrature(a, b, epquad: real; var: abserr, result:
real);
    var i,n: integer;
        x,y,z,h:real;
{      k: integer      is a global variable denoting the rule used }
begin
    n:=1;
    repeat
        x:=a;
        h:=(b-a)/n;
        result := 0;
        for i:=1 to n do begin
              rule(x, x+h, k, z, y);
              result:=result+y;
              x:=x+h;
        end;
        abserr:=    a global estimate of the absolute value of the
                    total error
        n:=2*n
    until abserr < epquad
end;
```

Let us discuss the computational cost of this method for a $C^s[a, b]$, $s \geq 1$, function. Let $r - 1$ be the degree of precision of the rule $\mathfrak{Q}$, $q = \min[r, s]$; let v_0 and v_1 be the evaluation coefficients for the rule $\mathfrak{Q}$ and let 2^k be the number of subintervals. The number of evaluations is

$$N = 2^k v_1 + v_0. \tag{3.3}$$

Applying relation (A.1) of the Appendix to each subinterval of length $h = 2^{-k}(b - a)$ we get

$$|I - I(N)| \leq (b - a)^{q+1} 2^{-q(k+1)-1} c_q \max_{x \in [a,b]} \frac{|f^{(q)}(x)|}{q!},$$

where c_q is a constant related to the Peano kernel (see Davis and Rabinowitz, (1984)). Moreover,

$$E = -\log_{10}|I - I(N)| \geq kq \log_{10} 2 + C_1, \qquad C_1 \text{ independent of } k. \tag{3.4}$$

Combining (3.3) and (3.4) we get

$$N = O(10^{E/q}).$$

This result holds also for functions in the classes $\mathcal{A}[a, b, \delta]$ or $\mathcal{M}[a, b, s]$, $s \geq 1$, with $q = r$ or $q = \min[r, s]$, respectively.

A more accurate result can be derived for a symmetric interpolatary formula $\mathfrak{Q}$ and a $C^{(s)}[a, b]$ function $f(x)$, $s \geq r + 1$. By applying to each subinterval Theorem 3.3 of Favati *et al.* (1992), which gives an asymptotically exact representation of the error, we get

$$I - I(N) = \sum_{\substack{j=r \\ j \text{ even}}}^{t} \left(\frac{h}{2}\right)^{j+1} \frac{\beta_j}{j!} \sum_{i=1}^{2^k} f^{(j)}\left(a + \left(i - \frac{1}{2}\right) h\right) + O(h^{t+1}),$$

where $r \leq t \leq s - 1$ is an even nonnegative integer and, for j even, β_j is the error of integrating on $[-1, 1]$ the polynomial x^j by using rule $\mathfrak{Q}$.

From the asymptotic formulas for the midpoint rule we can write

$$h \sum_{i=1}^{2^k} f^{(j)}\left(a + \left(i - \frac{1}{2}\right) h\right) = \sum_{\substack{m=j \\ m \text{ even}}}^{t} h^{m-j} \frac{C_{m-j}}{(m-j)!} \left(f^{(m-1)}(b) - f^{(m-1)}(a)\right) + O(h^{t-j+2}),$$

where $C_0 = 1$ and C_i, i even, is defined in Davis and Rabinowitz (1984, p. 139). Then we have

$$I - I(N) = \sum_{\substack{j=r \\ j \text{ even}}}^{t} 2^{-j-1} \frac{\beta_j}{j!} \sum_{\substack{m=j \\ m \text{ even}}}^{t} h^m \frac{C_{m-j}}{(m-j)!} \left(f^{(m-1)}(b) - f^{(m-1)}(a)\right) + O(h^{t+1}),$$

If $f^{(r-1)}(a) \neq f^{(r-1)}(b)$, by setting $t = r$ we get

$$I - I(N) = 2^{-r-1} \frac{\beta_r}{r!} h^r \left(f^{(r-1)}(b) - f^{(r-1)}(a)\right) O(h^{r+1}).$$

More generally, let p be the smaller even integer greater than $r - 1$ for which $f^{(p-1)}(a) \neq f^{(p-1)}(b)$. Setting $t = p$, we get

$$I - I(N) = h^p(f^{(p-1)}(b) - f^{(p-1)}(a)) \sum_{\substack{j=r \\ j \text{ even}}}^{p} 2^{-j-1} \frac{\beta_j}{j!} \frac{C_{p-j}}{(p-j)!} + O(h^{p+1})$$
$$= 2^{-kp}(b-a)^p C_{r,p}[f^{(p-1)}(b) - f^{(p-1)}(a)] + O(2^{-k(p+1)}), \tag{3.5}$$

where

$$C_{r,p} = \sum_{\substack{j=r \\ j \text{ even}}}^{p} 2^{-j-1} \frac{\beta_j}{j!} \frac{C_{p-j}}{(p-j)!}.$$

Therefore,

$$10^{-E} \sim 2^{-kp}|(b-a)^p C_{r,p}[f^{(p-1)}(b) - f^{(p-1)}(a)]|$$

and

$$N \sim v_1(b-a)|C_{r,p}[f^{(p-1)}(b) - f^{(p-1)}(a)]|^{1/p} 10^{E/p}. \tag{3.6}$$

For the PNAQ algorithm the density of the function evaluations is constant in the integration interval; thus the grey points always lie on a horizontal line. If the integrand function is regular enough, the black points also lie on a flat curve. As an example, the behaviour of PNAQ for the function $\sin(x)$, integrated on $[-1, \pi]$, is shown in Fig. 3.1.

On the other hand, if the function presents some difficulties, the distribution of black points has minima corresponding to the integration difficulties. The example in Fig. 3.2 shows the behaviour of PNAQ, after a

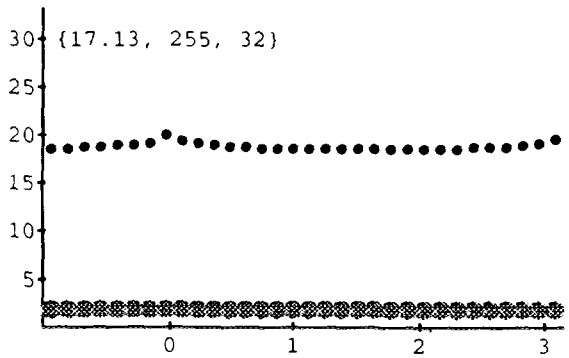

FIG. 3.1. Sine function, Clenshaw–Curtis 7-point, 32 subintervals. The meaning of symbols and axes is the same as Fig. 2.3.

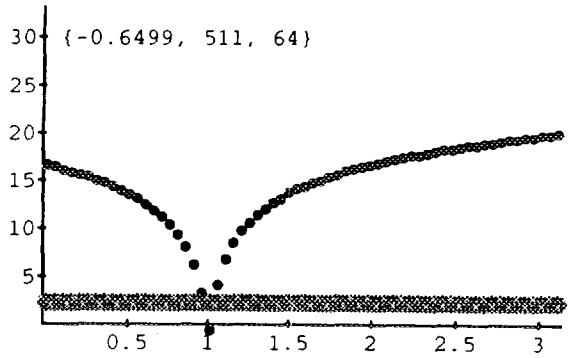

FIG. 3.2. Peak function, Clenshaw–Curtis 7-point, 64 subintervals. The meaning of symbols and axes is the same as Fig. 2.3.

subdivision into 64 subintervals, for the function $(10^{-4} + (x - 1)^2)^{-1}$, integrated on $[0, \pi]$, which presents a large peak at $x = 1$ (and complex poles at $z = 1 \pm i/100$).

Double Exponential Quadrature

A common device to integrate functions that are unbounded at one or both ends of the integration interval is to apply a variable transformation. The mapping function is so chosen that the singular points of the integrand are moved to infinity, converting the improper integral into a convergent infinite one, and the midpoint rule is applied to the transformed integral. In other words, after the transform

$$I = \int_{-1}^{1} f(x)\,dx = \int_{-\infty}^{\infty} f(\phi(u))\,\phi'(u)\,du,$$

I is approximated by

$$\int_{-A}^{A} f(\phi(u))\,\phi'(u)\,du.$$

Then, applying the midpoint rule, the finite sum

$$h \sum_{j=-n}^{n} f(\phi(jh))\,\phi'(jh), \qquad h(n + \tfrac{1}{2}) = A \tag{3.7}$$

is computed. The analytical error of this method is the sum of two components, one deriving from approximating the infinite integral in a finite

interval (trimming error) and the other is due to the use of the midpoint formula (discretization error). Among the various transforms proposed in the literature, the best one is the so-called double exponential, i.e., $\phi(u) = \tanh(\pi/2 \sinh u)$, which causes the transformed integrand to decay as the double exponential function $\exp(-\pi/2 \exp |u|)$.

The trimming error is an asymptotically decreasing function of $|A|$ exhibiting a double exponential decay; on the other hand, the discretization error decreases in a transitory phase and tends to an asymptotic value which is studied by letting $A = +\infty$. For any choice of h and A, an open symmetric quadrature formula (not of interpolatory type) in the interval $(-1, 1)$ can be derived. The use of such a rule will be called, in the following, the DE-quadrature.

The relation between the number of evaluations and the error has been studied for $\mathcal{ES}$ functions (Takahasi and Mori, 1974), for further references see Davis and Rabinowitz (1984, p. 214). In the suggested algorithm h is reduced and $|A|$ is increased until the desired accuracy is reached. The resulting error is estimated asymptotically as

$$\exp\left(-C \frac{N}{\log N}\right),$$

that is, in our notation

$$N = O(E \log E).$$

To implement the method in practice, one has to empirically estimate both the trimming and discretization errors, stopping the process when the desired accuracy is reached. The sum (3.7) is evaluated with increasing values of n and two different values of h (e.g., 2^{-k} and 2^{-k-1}). The difference between the resulting approximations of the integral (with the same n) gives an estimate of the discretization error. Since the transformed integrand has a double exponential decay, when the contribution of further terms in (3.7) becomes negligible with respect to the discretization error, the step size is halved (i.e., k is increased by one) and the process starts again. Due to the properties of trapezoidal sums, no function values are wasted.

Double exponential quadrature is implemented as one of the possible methods in the built-in *Mathematica*™ NIntegrate routine. Clearly this method has a poor behaviour for integrands with singularities inside the integration interval.

Remark. In the literature there exist proofs of lower bounds for the possible asymptotic behaviour of the errors involved in a quadrature scheme that are higher than the upper bound proved for the DE-quadra-

ture; see Stenger (1978) and Traub *et al.* (1988). Murota and Iri (1982) note that there is no contradiction because the classes of functions examined by Stenger are different.

4. Global Adaptive Quadrature Schemata

Global Adaptive Composition of a Fixed Rule

The first schema of automatic adaptive quadrature that we consider (called GQ in the following) is essentially the algorithm used in QAG (Piessens *et al.*, 1983), and QXG (Favati *et al.*, 1991b) and consists of repeatedly dividing the worst subinterval until the total error estimate is better than the user requested tolerance. The relevant information on the subintervals is stored in a priority queue, ordered according to the error estimate and accessed by the routines `put_interval` and `get_interval`. We assume that the routine

```
procedure rule(a, b:real; k: integer; var abserr, result:
                real)
```

implements a symmetric interpolatory rule $\mathcal{Q}$ with degree of precision $r - 1$ and evaluation coefficients v_0 and v_1.

Algorithm 4.1.

```
procedure GQuadrature (a,b,epquad: real; var abserr,result: real);
    var res,est,res1,est1,res2,est2,middle: real;
{       k: integer    is a global variable denoting the rule used }
begin
    rule(a,b,k,abserr,result);                { integrate on the first
                                                interval }
    put_interval(a,b,abserr,result);          { put the interval in the
                                                queue }
    while abserr > epquad do begin
         get_interval(a,b,est,res);           { get the interval with
                                                the largest error }
         middle:=(a+b)/2          { perform bisection }
         rule(a,middle,k,est1,res1);
         rule(middle,b,k,est2,res2);
         put_interval(a,middle,est1,res1);
         put_interval(middle,b,est2,res2);
         result:=result-res+res1+res2;        { update estimates }
         abserr:=abserr-est+est1+est2;
    end;
end;
```

Let m be the number of subintervals at some step of the integration process and let $e^{(i)}$, $i = 1, 2, \ldots, m$ be the true errors associated to the subintervals. In this algorithm abserr is an approximation of the quantity $\sum_{i=1}^{m} |e^{(i)}|$ which, in general, is greater than or equal to $|I - I_{\text{computed}}|$. There are some special cases where the sum of the absolute values of the errors on the subintervals is much greater than the total error $|I - I_{\text{computed}}|$. In these cases, the algorithm GQ can be outperformed by the algorithm PNAQ (e.g., it is well known that the best way to integrate periodic functions is to use trapezoidal or midpoint rules).

If f is a polynomial of degree less than r, only one interval suffices to exactly compute the integral, and the computational cost is $O(1)$. Otherwise, by using Assumption 2.2 one has abserr $= \sum_{i=1}^{m} |e^{(i)}|$, and the asymptotic computational cost is studied by investigating the relationship between the number of function evaluations N and the quantity $E = -\log_{10}$ abserr in the infinite integration process (using exact real arithmetic) produced by a zero error requirement. The following facts can be easily verified:

(i) any subinterval with nonzero error will be halved in the integration process;

(ii) subintervals with exactly zero error do not further influence the integration process;

(iii) the number of subintervals increases by one at each bisection.

When a global adaptive quadrature algorithm, based on bisections, is used, then, after some transitory phase, the integration process can be reduced to independent integrations of the same function on different intervals, on which the integrand function has no more than one singularity. Moreover, any point of the form $a + (b - a)\gamma$, where γ is a binary machine number, will be the end-point of two subintervals at some step of the integration process. In the following such points will be called reachable. Step singularities (of the function or its derivatives), located at reachable points, will be automatically removed. More precisely, we say that a function of class $\mathcal{M}$ has only apparent singularities if there exist $h + 1$ reachable points $a = x_0 < x_1 < \cdots < x_h = b$, such that the integrand is of class $\mathcal{A}[x_i + \delta_i, x_{i+1} - \delta_i, \delta_i]$, $i = 0, 1, \ldots, h - 1$.

Our goal is to determine both the order of $N(E)$ and the constant hidden by the "O" notation. In the following we prove that $N = \Theta(10^{E/r})$ for several classes of functions with no more than one singularity in the integration interval, and we determine the expression of the constant involved. Then, we will show that the same bound, with the same expression of the constant, holds for a generic $\mathcal{M}[a, b, s]$ function, with one additional condition on the growth of rth derivative. Numerical experiments suggest the conjecture that this additional condition is not needed.

Analysis 1: C^{r+2} Functions

Let us discuss the computational cost of GQ for $C^{r+2}[a, b]$ functions. The following analysis is applicable to functions in the classes $\mathcal{A}[a, b, \delta]$ or $\mathcal{M}[a, b, s]$, with $s \geq r + 2$, as well. We denote with $\mathcal{R}(m)$ the set of the m subintervals in the queue at the mth step of the integration process. Let $R = [x - \lambda, x + \lambda] \in \mathcal{R}(m)$ be a subinterval of length 2λ with a nonzero associated error e, let $R_1, R_2 \in \mathcal{R}(m + 1)$ be the subintervals obtained by halving R and let e_1, e_2 be the errors of the integrals of f on R_1 and R_2, respectively. In virtue of Theorem A.1, since $|f^{(r+2)}(x)|$ is bounded on $[a, b]$, we have, apart from numerical coincidences,

$$e \sim \lambda^{r+1} f^{(r)}(x) \frac{\beta_r}{r!}. \tag{4.1}$$

Then, from Theorem A.4, we have

$$e_1 = 2^{-(r+1)}e + O(\lambda^{r+2}), \qquad e_2 = 2^{-(r+1)}e + O(\lambda^{r+2}). \tag{4.2}$$

The above equations mean that, when a sufficiently small subinterval is halved, two subintervals are generated for which the error is reduced by a factor close to $2^{-(r+1)}$.

Let the integration process have progressed enough to have m_0 subintervals sufficiently small to neglect the $O(\lambda^{r+2})$ terms in (4.2). Let $e^{(i)}$, $i = 1, 2, \ldots, m_0$, be the errors associated to the subintervals and let P be the best subinterval (with nonzero error) in $\mathcal{R}(m_0)$, i.e.,

$$|e_P| = \min_{\substack{1 \leq i \leq m_0 \\ e^{(i)} \neq 0}} |e^{(i)}|.$$

Let m_1 be the number of subintervals when P is halved. Any other subinterval in $\mathcal{R}(m_1)$ has been generated by bisecting a subinterval having an error with absolute value greater than $|e_P|$. Then, applying Theorem A.4, for any interval $R \in \mathcal{R}(m_1)$ of length λ we have

$$2^{-(r+1)}|e_P|(1 + O(\lambda)) \leq |e_R| \leq |e_P|. \tag{4.3}$$

Let

$$e_{\text{MIN}} = \min_{\substack{1 \leq i \leq m \\ e^{(i)} \neq 0}} |e^{(i)}|, \qquad e_{\text{MAX}} = \max_{1 \leq i \leq m} |e^{(i)}|;$$

then, for $m = m_1$ inequality (4.3) implies that

$$e_{\text{MAX}} \leq 2^{r+1} e_{\text{MIN}}(1 + O(\lambda)).$$

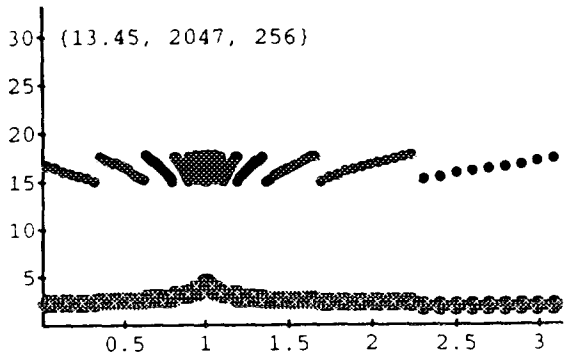

FIG. 4.1. Peak function, Clenshaw–Curtis 7-point, 256 subintervals. The meaning of symbols and axes is the same as Fig. 2.3.

This relation can be proved for any successive step of the integration process by induction on the number m of subintervals starting from m_1 and using relation (4.2). This allows us to write

$$e_{\mathrm{MAX}} \gtrsim 2^{r+1} e_{\mathrm{MIN}}. \tag{4.4}$$

Relation (4.4) implies that, in the graphical representation explained in Fig. 2.3, most black points lie in a strip of size $(r + 1) \log_{10} 2$ decimal digits. As an example, Fig. 4.1 shows the behaviour of GQ for the function $(10^{-4} + (x - 1)^2)^{-1}$ after 255 interval bisections. A comparison with Fig. 3.2 shows that the adaptivity achieves a more uniform error distribution at the expense of a nonuniform evaluation density.

Let us consider a point u belonging to the integration interval. Let $2\lambda(u)$ and $e(u)$ denote the length and the associated error of the subinterval containing the point u at some step of the integration process, respectively. Relation (4.1) can be written as

$$e(u) \sim \lambda(u)^{r+1} f^{(r)}(y(u)) \frac{\beta_r}{r!},$$

where $y(u)$ denotes the midpoint of the subinterval which contains u. Since $y(u) \to u$ as $\max_{u \in [a,b]} \lambda(u) \to 0$, then we can write

$$e(u) \sim \lambda(u)^{r+1} f^{(r)}(u) \frac{\beta_r}{r!}, \qquad \text{as } \max_{u \in [a,b]} \lambda(u) \to 0$$

and, therefore,

$$\lambda(u) \sim \left[|e(u)| \Big/ \left(|f^{(r)}(u)| \frac{|\beta_r|}{r!} \right) \right]^{1/(r+1)}.$$

The density of subintervals is $(2\lambda(u))^{-1}$ and the number m of subintervals contained in $T = [a, b]$ is

$$m = \frac{1}{2} \int_T \lambda(u)^{-1}\, du \sim \frac{1}{2} \int_T \left(|f^{(r)}(u)| \frac{|\beta_r|}{r!} \Big/ |e(u)| \right)^{1/(r+1)} du \leq C_T e_{\text{MIN}}^{-1/(r+1)}, \tag{4.5}$$

where

$$C_T = \frac{1}{2} \left(\frac{|\beta_r|}{r!} \right)^{1/(r+1)} \int_T |f^{(r)}(u)|^{1/(r+1)}\, du \tag{4.6}$$

is a constant depending on $\mathfrak{Q}$, f, and T. Since

$$10^{-E} = \texttt{abserr} = \sum_{i=1}^{m} |e^{(i)}| = \frac{1}{2} \int_T |e(u)| \lambda(u)^{-1}\, du,$$

we have

$$10^{-E} \sim \frac{1}{2} \int_T |e(u)|^{r/(r+1)} \left(|f^{(r)}(u)| \frac{|\beta_r|}{r!} \right)^{1/(r+1)} du \leq C_T e_{\text{MAX}}^{r/(r+1)}, \tag{4.7}$$

Combining (4.5) and (4.7) and using $N = v_1 m + v_0$ and Lemma A.2 of the Appendix, we get

$$v_1 C_T^{(r+1)/r} 10^{E/r} \lesssim N \lesssim \left(\frac{e_{\text{MAX}}}{e_{\text{MIN}}} \right)^{1/(r+1)} v_1 C_T^{(r+1)/r} 10^{E/r}. \tag{4.8}$$

Finally, using (4.4), relation (4.8) can be rewritten as

$$v_1 C_T^{(r+1)/r} 10^{E/r} \lesssim N \tag{4.9a}$$

$$N \lesssim 2 v_1 C_T^{(r+1)/r} 10^{E/r}, \tag{4.9b}$$

i.e.,

$$N = \Theta(10^{E/r}).$$

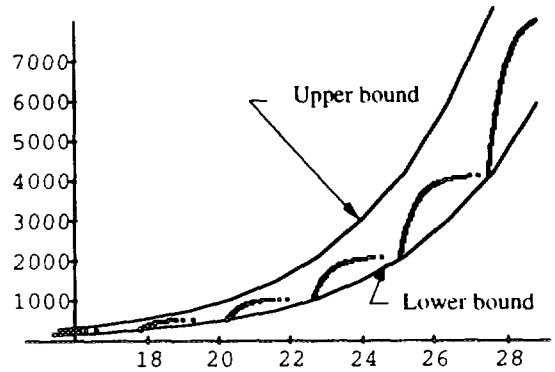

FIG. 4.2. N versus E for the program GQ using Clenshaw–Curtis 7-point rule, applied to the integrand x^8, compared with the theoretical bounds given by (4.9).

In the special case $f(x) = x^r$, the bound (4.9a) and the expression (3.6) for PNAQ are both equal to

$$v_1 \left(\frac{b-a}{2}\right)^{(r+1)/r} |\beta_r|^{1/r} 10^{E/r}.$$

The computational cost of GQ for the function x^8, integrated on [0, 1] with the Clenshaw–Curtis 7-point rule is presented in Fig. 4.2 compared with the theoretical bounds of (4.9). We note that, since the rth derivative of x^8 is a constant, the lower bound is periodically reached when all the intervals have the same length and the algorithm coincides with PNAQ. The computational cost of a nonpolynomial function like $\exp(20\,x)$, integrated on [0, 1], is less oscillating (Fig. 4.3).

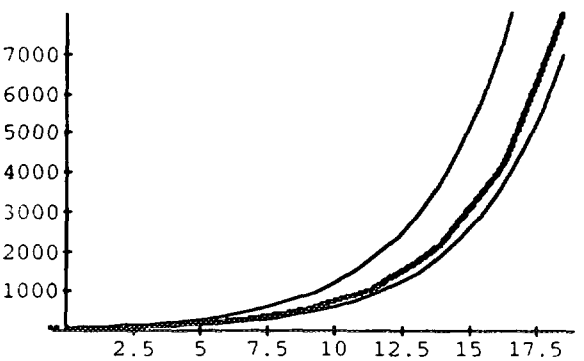

FIG. 4.3. N versus E for the program GQ using Clenshaw–Curtis 7-point rule, applied to the integrand $\exp(20\,x)$, compared with the theoretical bounds given by (4.9).

Analysis 2: Bounded $(r + 2)$ th Derivative Integrands

We now investigate the computational cost of the global adaptive algorithm for a function in the class $\mathcal{M}[a, b, s]$ with $-1 \leq s < r + 2$ and having the $(r + 2)$th derivative bounded in $[a, b]$. We assume also that only one singular point $a \leq y \leq b$ is present. We divide the subintervals contained in the queue, at any time of the integration process, into three classes:

(1) the "red" subinterval which contains the singularity,

(2) the "white" subintervals,

(3) the "black" subintervals.

The definition of the color of a subinterval is recursive:

(i) at the beginning of the integration process there is only the red subinterval: $[a, b]$;

(ii) the red subinterval is bisected into one red subinterval (containing the singularity) and one white subinterval;

(iii) a white subinterval is bisected into two black subintervals;

(iv) a black subinterval is bisected into two black subintervals.

The meaning of white and black subintervals is clear in the following graph taken after 127 bisections for the step function

$$\begin{cases} \sin 10\, x, & x \leq 1, \\ 2 \exp(x), & x > 1, \end{cases}$$

integrated on $[0, \pi]$.

We denote with $S(i)$ the red subinterval after i bisections. As long as the error $e_{S(i)}$ on the subinterval $S(i)$ is dominating, the red subinterval is divided. If the error on $S(i)$ is less than the error on some white subinterval (say W), then W is divided and black subintervals start to be produced. The integration on black subintervals will proceed without being affected by the singularity.

The following lemma allows us to relate the number of bisections of the interval containing the singularity with the quantity e_{MAX}, which denotes the error of the worst interval at the same time. Note that, during the integration process e_{MAX} is attained repeatedly on red, white, and black intervals, so it is not worth distinguishing the color of the worst interval. On the other hand for e_{MIN} on the whole interval a lower bound cannot be derived (very small values can be attained on white intervals, see Fig. 4.4). We consider, instead, the quantity e_{MINB} that denotes the minimal error on black intervals and for which a relation analogous to (4.4) holds.

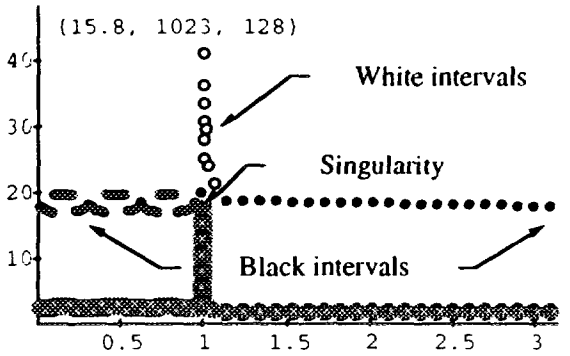

FIG. 4.4. Step function. Clenshaw–Curtis 7-point, 128 subintervals. The meaning of symbols and axes is the same as Fig. 2.3; moreover, the open circles denote the negative base-10 logarithm of the error for white intervals.

LEMMA 4.1. *The number i of bisections performed to get the red subinterval $S(i)$ is related to e_{MAX} by the following bound:*

$$i \leq -\log_2 e_{\mathrm{MAX}} + \log_2 D.$$

Proof. From Theorem A.2 of the Appendix we have $|e_{S(i)}| \leq B(b - a)2^{-i-1}$, i.e.,

$$i \leq -\log_2 |e_{S(i-1)}| + \log_2 B(b - a).$$

Since $|e_{S(i-1)}| \geq e_{\mathrm{MAX}}$, the thesis is proved with $D = B(b - a)$. ∎

Let T be the subset of the integration interval covered by black subintervals and let m_{B}, m_{W} be the numbers of black and white subintervals, respectively. During the integration process $\max_{u \in T} \lambda(u) \to 0$ and $T \to [a, b]$. Since the first $r + 2$ derivatives of f are bounded in $[a, b]$ and black subintervals are obtained by halving subintervals that do not contain the singularity, we have

(i) $C_{[a,b]}$, defined in (4.6), has a finite value;

(ii) on the set T, relation (4.5) holds for m_{B} and

$$C_T e_{\mathrm{MAX}}^{-1/(r+1)} \gtrsim m_{\mathrm{B}} \gtrsim C_T e_{\mathrm{MINB}}^{-1/(r+1)}. \tag{4.10}$$

Moreover, there are

$$m_{\mathrm{W}} \leq -\log_2 e_{\mathrm{MAX}} + \log_2 D$$

white subintervals and, since $m = m_B + m_W + 1$ and $-\log e_{MAX}$ is dominated by $e_{MAX}^{-1/(r+1)}$, then $m \sim m_B$.

In a similar way it is possible to prove the relation

$$10^{-E} = \text{abserr} = \sum_{i=1}^{m} |e^{(i)}| \sim \frac{1}{2} \int_T |e(u)|\lambda(u)^{-1}\, du,$$

that ensures that the contribution of white and red intervals to the total error is negligible. Finally relation (4.9) can be obtained again for $T \to [a, b]$ and the computational cost is the same as in the regular case.

It is possible to verify this fact, experimentally, for two functions which have the same constant $C_{[-1,\pi/2]}$:

$$\text{F1: } \exp(x); \qquad \text{F2: } \begin{cases} -\exp(x), & x \le 0 \\ \exp(x), & x > 0 \end{cases} \text{ integrated on } [-1, \pi/2].$$

Both the functions have been integrated with GQ (using Gauss–Legendre 5-point rule) until 42 decimal digits of accuracy have been obtained. As a matter of fact, after an initial transitory phase in which the smooth function F1 is integrated much better, both the complexity plots remain between the theoretical limits of (4.9).

In the case of piecewise polynomials of degree less than r, there are no black subintervals since the error on white subintervals is zero. In this case the only source of error is the singularity. It is easy to prove that the complexity function is linear, i.e., $N = \Theta(E)$. It is remarkable that the linear computational cost is attained by GQ for any singular function in the transition phase before black subintervals begin to be generated.

Analysis 3: Unbounded Integrands with an End-point Singularity

Let us now investigate the computational cost of the global adaptive algorithm restricting the analysis to an $\mathcal{M}[0, 1, -1]$ function $f(x)$ with only one unbounded singularity at 0 and

$$\alpha = \inf\{\beta: |f(z)| = O(|z|^{-\beta}),\ z \in \Gamma(\delta_1, \delta_2, \delta_1)\} < 1.$$

We assume also that f satisfies the additional property (2.1). We divide the subintervals contained in the queue, at any time of the integration process, into red, white, and black subintervals, as in the previous analysis. For a fixed constant $\varepsilon > 0$, an interval $[x - \lambda, x + \lambda]$ is called "large" if $\lambda \ge (x + \lambda)^{1+3\varepsilon}$, and "small" otherwise.

The following lemma allows us to relate the number of bisections of the interval containing the singularity with the quantity e_{MAX}, which denotes the error of the worst interval at the same time.

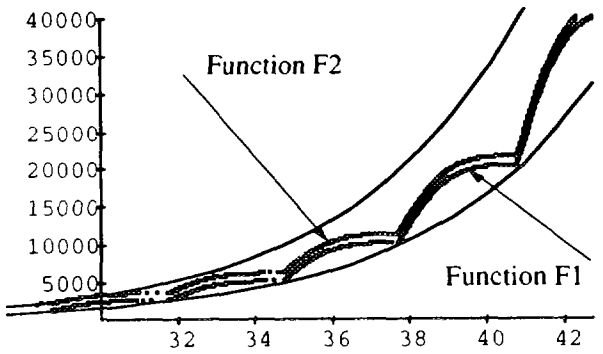

FIG. 4.5. N versus E $(30 < E < 42)$ for the program GQ using Gauss–Legendre 5-point rule, applied to the integrands F1 and F2, compared with the theoretical bounds given by (4.9).

LEMMA 4.2. *The number i of bisections performed to get the red subinterval $S(i)$ is related to e_{MAX} by the following bound:*

$$i \leq \frac{-\log_2 e_{\mathrm{MAX}} + \log_2 B}{1 - \alpha - \varepsilon}$$

Proof. The proof is analogous to that of Lemma 4.1, using Theorem A.3 instead of Theorem A.2. ■

Therefore the number of white subintervals is $m_W = O(-\log e_{\mathrm{MAX}})$. If we take $\varepsilon < (1 - \alpha)/(3r + 4)$ and $t = 2^{-i}$, then from Lemma A.3, the number of large subintervals m_L contained in $[t, 1]$ is

$$m_L = O(t^{-3\varepsilon}) = O(2^{3\varepsilon i}) = O(e_{\mathrm{MAX}}^{-3\varepsilon/(1-\alpha-\varepsilon)} = o(e_{\mathrm{MAX}}^{-1/(r+1)}).$$

The analysis of the asymptotic computational cost can be carried out by using a technique similar to that used for the previous case, considering black small subintervals instead of black subintervals.

Let T be the subset of the integration interval covered by black small subintervals, during the integration process $\max_{u \in T} \lambda(u) \to 0$ and $T \to [0, 1]$. From Lemma A.1, we have $|f^{(r)}(x)|^{1/(r+1)} = O(x^{-(\alpha+\varepsilon+r)/(1+r)})$, and $C_{[0,1]}$, defined in (4.6), has a finite value. Relation (4.10) is true for the number of small black intervals m_{BS} and relation (4.7) holds for the error on T. Since $m \leq m_{\mathrm{BS}} + m_{\mathrm{w}} + m_{\mathrm{L}} + 1$, then $m \sim m_{\mathrm{BS}}$. Applying Theorem A.5 to the small black subintervals we can derive again relation (4.4) on T and relation (4.9) can be obtained again for $T \to [0, 1]$. This result has been experimentally verified for the function $x^{-1/2} \log x$, integrated on $[0, \pi]$ (Fig. 4.6).

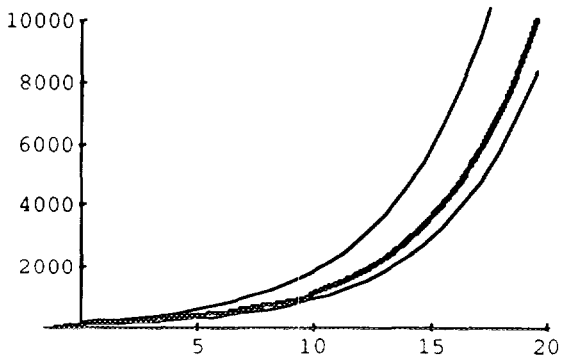

FIG. 4.6. N versus E for the program GQ using Gauss–Legendre 5-point rule, applied to the function $x^{-1/2} \log x$, compared with the theoretical bounds given by (4.9).

It is interesting to verify experimentally the behaviour of the singular unbounded function F3: $x^{-1/4}(3/4 \cos \log x - \sin \log x)$, integrated on [0, 1], which does not satisfy property (2.1). Note that F3 changes sign infinitely many times in [0, 1]. Also in this case the theoretical bounds bracket the complexity; it can be conjectured that hypothesis (2.1) is not needed for the validity of (4.9) (Fig. 4.7).

Analysis 4: Unbounded $(r + 2)$th Derivative Integrands

Let us now investigate the computational cost of the global adaptive algorithm for a bounded $\mathcal{M}[a, b, s]$ function $f(x)$ with $-1 \leq s < r + 2$, and only one singular point $a \leq y \leq b$. Let

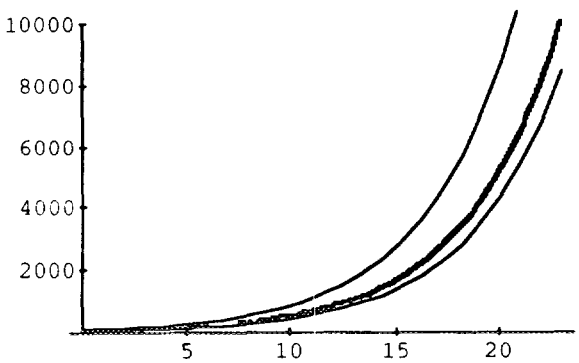

FIG. 4.7. N versus E for the program GQ using Gauss–Legendre 5-point rule, applied to the integrand F3, compared with the theoretical bounds given by (4.9).

$$\alpha_1 = \inf\{\beta : |f(z)| = O(|z - y|^{-\beta}),\ z \in \Gamma(y - \delta_2,\ y - \delta_1,\ \delta_1)\} < 1,$$

$$\alpha_2 = \inf\{\beta : |f(z)| = O(|z - y|^{-\beta}),\ z \in \Gamma(y + \delta_1,\ y + \delta_2,\ \delta_1)\} < 1.$$

We assume also that f satisfies the additional property (2.1). We divide the subintervals contained in the queue, at any time of the integration process, into three classes: red, white, and black subintervals. We associate a color to each subinterval recursively:

(i) at the beginning of the integration process there is only the red subinterval;

(ii) the red subinterval is bisected into one red subinterval (that containing the singularity) and a white subinterval;

(iii) a white subinterval is divided into two black subintervals if neither of them is adjacent to the interval containing the singularity; otherwise it is split into one black and one white subinterval (the one adjacent to the interval containing the singularity);

(iv) a black subinterval is bisected into two black subintervals.

Moreover, we assume that the red interval is not larger than any other white subinterval.

For a fixed constant $\varepsilon > 0$, an interval $[x - \lambda,\ x + \lambda]$ is called "large" if $\lambda \geq (|x - y| + \lambda)^{1+3\varepsilon}$, and "small" otherwise. From Lemma 4.1 the number of white subintervals is $m_W = O(-\log e_{MAX})$. The red interval is surrounded by two white intervals of length not smaller than $2^{-i(b-a)}$. The number of large subintervals m_L contained in $[a, b]$ can be bounded taking $\varepsilon < 1/(3r + 3)$ and applying Lemma A.3 to the intervals $[a, y]$ and $[y, b]$ with $|t - y| > 2^{-i(b-a)}$:

$$m_L = O(|t - y|^{-3\varepsilon}) = O(2^{3\varepsilon i}) = O(e_{MAX}^{-3\varepsilon}) = o(e_{MAX}^{-1/(r+1)}).$$

The analysis of the asymptotic computational cost can be carried out by using a technique similar to that used for the previous case. Let T be the subset of the integration interval covered by black small subintervals, during the integration process $\max_{u \in T} \lambda(u) \to 0$ and $T \to [a, b]$. Relation (4.10) is true for the number of small black intervals m_{BS} and relation (4.7) holds for the error on T. Since $m \leq m_{BS} + m_W + m_L + 1$, then $m \sim m_{BS}$. Applying Theorem A.5 to the small black subintervals we can derive again relation (4.4) on T and relation (4.9) can be obtained again for $T \to [a, b]$.

Analysis of GQ for Integrands with Multiple Singularities

Now we have the theoretical background to study the total computational cost of the algorithm GQ for the integration of a generic function in $\mathcal{M}[a, b, s]$.

LEMMA 4.3. *Consider an integrable function f and a set of disjoint intervals $T_1, T_2, \ldots, T_k$. If the following relations hold on each T_j,*

$$m_j \sim \frac{1}{2}\int_{T_j}\left(|f^{(r)}(u)|\,\frac{|\beta_r|}{r!}\Big/|e(u)|\right)^{1/(r+1)} du,$$

$$10^{-E_j} \sim \frac{1}{2}\int_{T_j}|e(u)|^{r/(r+1)}\left(|f^{(r)}(u)|\,\frac{|\beta_r|}{r!}\right)^{1/(r+1)} du,$$

then the total computational cost on $T = T_1 \cup T_2 \cup \cdots \cup T_k$ satisfies Eq. (4.9) with

$$C_T = \frac{1}{2}\left(\frac{|\beta_r|}{r!}\right)^{1/(r+1)}\int_{T_1\cup T_2\cup\cdots\cup T_k}|f^{(r)}(u)|^{1/(r+1)}\,du = \sum_{j=1}^{k} C_{T_j}.$$

Proof. Since the number of intervals and the absolute value of the error are additive quantities, relations (4.5)–(4.9) can be derived again. ■

THEOREM 4.2. *For a function f in $\mathcal{M}[a, b, s]$ with singular points $y_1, y_2, \ldots, y_k$ and satisfying the additional property* (2.1), *the total asymptotic computational cost of the algorithm GQ satisfies Eq.* (4.9) *with*

$$C_T = \frac{1}{2}\left(\frac{|\beta_r|}{r!}\right)^{1/(r+1)}\int_a^b |f^{(r)}(u)|^{1/(r+1)}\,du, \qquad T = [a, b].$$

Proof. Let us split the interval $[a, b]$ into h subintervals $T_i = [x_i, x_{i+1}]$, $i = 0, 1, \ldots, h-1$, $x_0 = a$, $x_h = b$, such that

(i) each x_i is reachable.

(ii) each interval T_i contains no more than one singularity;

(iii) if a singularity is in a reachable point y then $y = x_i$ for some i.

The hypotheses of Lemma 4.3 can be fulfilled, by using for each interval T_i:

Analysis 1, if f is a $C^{r+2}[T_i]$ function;

Analysis 2, if f has a bounded $(r+2)$th derivative on T_i;

Analysis 3, if f is unbounded with singularity at a $(i = 1)$ or at b $(i = h)$;

Analysis 4, if f is bounded and has a singularity with unbounded $(r+2)$th derivative on T_i. ■

5. Double-Adaptive Quadrature Schemata

Overall Description of the Algorithm

From the results of the previous sections it turns out that the best performances for class $\mathcal{A}$ functions are obtained by the Clenshaw–Curtis quadrature ($N = O(E)$) and for general functions in class $\mathcal{M}$ they are obtained by the GQ algorithm ($N = \Theta(10^{E/r})$). In this section we introduce a quadrature schema that is applicable to functions of any class, attaining better complexity bounds than GQ.

The basic idea (see also Oliver, 1972) consists of combining, in a general global adaptive schema, the two main strategies for improving the approximation of the integral, i.e., the interval subdivision and the application of more accurate formulas. This idea is also the basis of the "h-p" version of the finite element method for the solution of partial differential equations (for a detailed analysis see Gui and Babuska, 1986).

In practice the active subintervals are ordered into a queue according to the error estimate, and at any step the subinterval with the worst error estimate can be bisected or processed with a higher degree formula (in the following we shall use formulas in the Clenshaw–Curtis family). The choice between the two alternatives is determined by the presence of difficulties in the subinterval. Obviously, if the location of the singularities is not a priori known, an empirical test (which can sometimes fail) has to be performed. For class $\mathcal{A}$ functions, since, apart from transitory numerical difficulties, no singularities can be detected, the integration process is reduced to the application of the Clenshaw–Curtis nonadaptive algorithm of Section 3 on a finite number of subintervals and the resulting computational cost is $O(E)$. In the following, we discuss, for integrands with only one singularity, the algorithm which uses the integration strategy of halving the worst subinterval, if it contains the singularity, and of processing it with a higher order formulas, otherwise. Without loss of generality, we assume that the integrand is continuable analytically in a sufficiently large neighborhood of the singularity. Finally, we present a general algorithm that can be applied to any function (including functions with several singularities).

Analysis 1: Unbounded Integrands with an End-point Singularity

We now investigate the computational cost of the double-adaptive algorithm for an $\mathcal{M}[0, 1 - 1]$ function $f(x)$, continuable analytically in the interior of $\Gamma(\frac{1}{2}, 1, \frac{1}{2})$, with only one unbounded singularity at zero and

$$\alpha = \inf\{\beta : |f(z)| = O(|z|^{-\beta}),\ z \in \Gamma(\tfrac{1}{2}, 1, \tfrac{1}{2})\} < 1.$$

FIG. 5.1. Subinterval structure.

Since the singularity is at zero, only the leftmost subinterval is bisected. At any step there is one "red" subinterval containing the singularity, which is halved more and more, generating "black" subintervals that never will be halved (Fig. 5.1).

Let $S^{(i)} = [0, 2^{-i}]$ be the red subinterval on which a Clenshaw–Curtis formula with m_0 nodes is applied. A bound on i as a function of the absolute value $\omega = e_{\mathrm{MAX}}$ of the error of the worst subinterval, is given in Lemma 4.2. Let $R^{(j)} = [2^{-j}, 2^{-j+1}]$ be a black subinterval and let $e^{(j)}(\mu)$ be the associated error when the Clenshaw–Curtis rule with $\mu \geq m_0$ nodes is applied on $R^{(j)}$. Let m_j be the number of nodes of the formula actually used on $R^{(j)}$. Let us assume that the integration process has progressed enough to guarantee that

$$m_h = \max_{0 \leq j \leq i} m_j > m_0.$$

From Theorem 3.1, with $\tau = \frac{1}{2}$, $\rho = 1 + \sqrt{2} > 2$, we obtain

$$|e^{(j)}(m_j)| \leq F\, 2^{-j-1} \max_{z \in \partial\Gamma(2^{-j}, 2^{-j+1}, 2^{-j-1})} |f(z)|\, \rho^{-m_j-2}, \qquad j = 1, 2, \ldots, i;$$

moreover, for any $0 < \varepsilon < 1 - \alpha$, there exist two constants Q and C_1 for which

$$\begin{aligned} |e^{(j)}(m_j)| &\leq FQ\, 2^{-j-1-(-j-1)(\alpha+\varepsilon)}\, \rho^{-m_j-2} < FQ\, 2^{-(1-\alpha-\varepsilon)(j+1)-m_j-2} \\ &< C_1\, 2^{-m_j}, \qquad j = 1, 2, \ldots, i. \end{aligned}$$

When the red subinterval is divided, for any $m_j > m_0, j = 1, 2, \ldots, i$, we have

$$|e^{(j)}(m_j)| \leq \omega \leq |e^{(j)}(m_j/2 + 1/2)|.$$

Then,

$$C_1\, 2^{-m_h/2-1/2} > |e^{(h)}(m_h/2 + 1/2)| > \omega$$

and

$$-\log_2 C_1 + m_h/2 + 1/2 < \log_2 1/\omega.$$

Therefore, there exist constants $C_2, \ldots, C_6$ for which, for ω sufficiently small,

$$m_h < C_2 \log_2 1/\omega + C_3;$$

$$N \leq i\, m_h + (i+1)\, m_0 < C_4 (\log_2 1/\omega)^2$$

$$10^{-E} \leq (i+1)\, \omega \leq C_5\, \omega \log_2 1/\omega$$

$$E \geq \log_{10} 1/\omega - \log_{10} \log_2 (1/\omega) - \log_{10} C_5 \geq C_6 \log_2 1/\omega$$

from which, finally, we have $N = O(E^2)$.

Analysis 2: Bounded Integrands with Internal Singularity

We now investigate the computational cost of the double-adaptive algorithm for a bound $\mathcal{M}[0, 1, s]$ function $f(x)$ with only one singular point $0 \leq y \leq 1$, continuable analytically in the interior of both $\Gamma(0, y/2, y/2)$ and $\Gamma((y+1)/2, 1, (1-y)/2)$. Let

$$\alpha_1 = \inf \{\beta : |f(z)| = O(|z - y|^{-\beta}),\ z \in \Gamma(0, y/2, y/2)\} < 1,$$

$$\alpha_2 = \inf \{\beta : |f(z)| = O(|z - y|^{-\beta}),\ z \in \Gamma((y+1)/2, 1, (1-y)/2)\} < 1.$$

There is one red subinterval containing the singularity, which is halved more and more, each time generating one red and one white subinterval. A white subinterval is divided into two black subintervals if neither of them is adjacent to the singularity; otherwise it is split into one black and one white subinterval. A Clenshaw–Curtis formula with m_0 nodes is applied on red and white subintervals. Black subintervals are processed with increasing order formulas and they are never halved. The structure of subintervals can be exemplified with the help of a binary tree: the root (level 0) denotes the interval $[0, 1]$; the 2^i nodes at level i denote the possible subintervals of length 2^{-i} (Fig. 5.2).

It is easy to see that there are only two white subintervals surrounding the red one. Let $S^{(i)}$ denote the red subinterval of length 2^{-i} $(i > 1)$; a bound on i as a function of $\omega = e_{\mathrm{MAX}}$ is given in Lemma 4.1. For what concerns black subintervals, we point out that the distance from the singularity of any black subinterval of length λ is greater than or equal to λ; moreover, if we assume that $S^{(i)}$ is not larger than any other subinterval, then there are exactly $3i - 5$ black subintervals. If the Clenshaw–Curtis rule with m_j nodes is used on the jth black subinterval, Theorem 3.1, with $\rho > 2$, can be applied. The analysis is similar to the previous one. Clearly, one has

$$|e^{(j)}(m_j)| < C_1\, 2^{-m_j}.$$

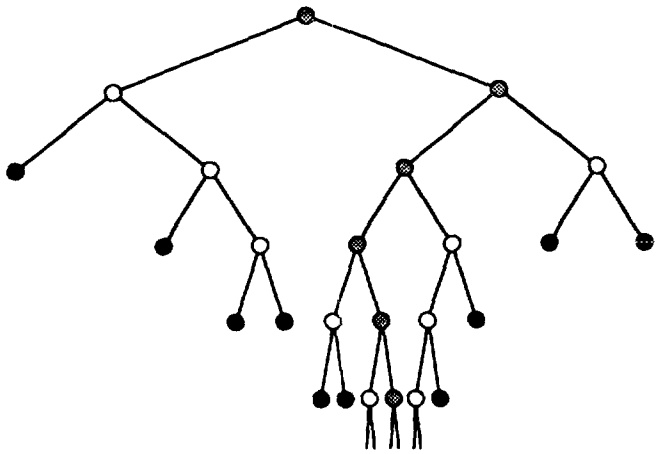

FIG. 5.2. Tree associated with the subinterval structure; the grey circles represent the red subintervals.

When the red subinterval is divided, for any $m_j > m_0, j = 1, 2, \ldots, 3i - 5$, we have

$$|e^{(j)}(m_j)| \leq \omega \leq |e^{(j)}(m_j/2 + 1/2)|.$$

Let $m_h = \max_{0 \leq j \leq 3i-5} m_j > m_0$, then

$$C_1 2^{-m_h/2-1/2} > |e^{(h)}(m_h/2 + 1/2)| \geq \omega.$$

Then, there exist constants $C_2, \ldots, C_6$ such that, for a sufficiently small ω,

$$\begin{aligned} m_h &< C_3 \log_2 1/\omega + C_2; \\ N &\leq (3\,i - 5)\, m_h + 3\,i\, m_0 < C_4 (\log_2 1/\omega)^2; \\ 10^{-E} &\leq (3i - 2)\, \omega \leq C_5\, \omega \log_2 1/\omega; \\ E &\geq \log_{10} 1/\omega - \log_{10} \log_2 (1/\omega) - \log_{10} C_5 \geq C_6 \log_2 1/\omega. \end{aligned}$$

Whence, finally, we have $N = O(E^2)$.

DAQ Algorithm

We now describe a general algorithm DAQ, implementing the double-adaptive strategy. DAQ also works with several singularities located anywhere in the interval. The program is based on the consideration that an

empirical test on fast convergence can detect the presence of both singularities and numerical difficulties that prevent the estimated error from behaving as predicted by the asymptotic theory.

By using the theory of the error for Clenshaw–Curtis quadrature for functions of class $\mathcal{A}$ and assuming that the bounds in relation (3.1) are asymptotically sharp, we get

$$e = |I - I(m)| \sim C_1 \rho^{-m-2},$$
$$e_1 = |I - I(2m-1)| \sim C_1 \rho^{-2m-1}.$$

Once given a constant $\beta \leq 2$,

$$\frac{e^\beta}{e_1} \sim C_1^{(\beta-1)} \rho^{(1-2\beta)} \rho^{(2-\beta)m} = C_2 \rho^{(2-\beta)m}.$$

I.e., the relation $\lim_{m\to\infty} (e_1/e^\beta) = 0$ is a necessary condition for the asymptotic validity of (3.1). Following these considerations we adopt the empirical test:

$$e_1 \leq 0.01\, e^{1.5}.$$

Algorithm 5.1 shows how DAQ can be implemented. The basic strategy is to apply bisection in the presence of integration difficulties and to apply the Clenshaw–Curtis rule if the convergence test has been passed. The function `test(est, est1)` returns true if $\texttt{est1} \geq 0.01\ \texttt{est}^{1.5}$ and is used to stop the use of more accurate rules and to force the application of bisection. Obviously, the test may fail, but the resulting overhead does not affect the asymptotic computational cost of the algorithm.

ALGORITHM 5.1.

```
procedure DAQuadrature (a, b, epquad: real; var abserr, result:
          real);
    var res,est,res1,est1,res2,est2,middle: real;
        k: integer;
{  k0: integer; is a global variable denoting the starting point
       for the local rule                                        }
{  procedure CCrule(a,b: real; k: integer; var abserr, result:
             real);                                             }
{       implements Clenshaw-Curtis Quadrature with 2^k+1 points  }
{    function test(est,est1 : real):boolean;                     }
{        tests the convergence after doubling the points         }
{    put_interval_k and get_interval_k handle the priority queue. }
{    k ≥ 0 means that the kth Clenshaw-Curtis rule has been
    applied                                                     }
{    k = -1 means that a singularity has been detected          }
```

```
begin
    CCrule(a,b,k0,abserr,result);          {integrate on the first
                                            interval}
    put_interval_k(a,b,k0, abserr, result); {put the interval in
                                             the queue}
    while abserr > epquad do begin
         get_interval_k(a,b,k,est,res);     {get the interval with
                                             the largest error}
         if k>=0 then

{ no singularity nor an integration difficulty was detected for the
  interval [a,b]  }
         begin
            k:=k+1;
            CCrule(a,b,k, est1, res1);
            result:=result-res+res1;            {update estimates}
            abserr:=abserr-est+est1;
            if test(est,est1) then k:=-1;       {test for convergence}
            put_interval_k(a,b,k,abserr,result)
         end else
{     a singularity or an integration difficulty was detected for
      the interval [a,b]                                          }
          begin
              middle:=(a+b)/2                  {perform bisection}
              CCrule(a,middle,k0,est1,res1);
              put_interval_k(a,middle,k0, est1, res1);
              CCrule(middle,b,k0,est2,res2);
              put_interval_k(middle,b,k0,est2,res2);
              result:=result-res+res1+res2;     {update estimates}
              abserr:=abserr-est+est1+est2
        end
    end;
end;
```

Since DAQ is a global adaptive quadrature algorithm based on bisections, after some transitory phase the integration process can be reduced to independent integrations on a finite number of intervals on which one of the following facts holds:

(1) the integrand is regular;

(2) the integrand has only one singularity and the interval is sufficiently small to satisfy the conditions used in Analyses 1 and 2.

Moreover, the considerations on reachable points and apparent singularities previously introduced apply too.

In practice, since the subintervals surrounding the singularity are bisected only if the proximity of the singularity affects the convergence of Clenshaw–Curtis rules, the complexity analysis made before, using black, white, and red intervals is conservative, then the overall computational cost is

$N = O(E)$ for functions in class $\mathcal{A}[a, b, \delta]$ and
for functions in class $\mathcal{M}[a, b, s]$ with only apparent singularities;

$N = O(E^2)$ for the other functions in class $\mathcal{M}[a, b, s]$.

Figures 5.3–5.5 show the behavior of GQ and DAQ compared for the function $x^{3/2}$ interated on [0, 1]. The reduced rate of growth of the number of subintervals in the DAQ algorithm is obvious.

A preliminary version of this algorithm was implemented in *Mathematica*™ high level language and compared with the straightforward use of the built-in (low level coded) *Mathematica*™ NIntegrate function. The new program is more efficient and reliable, than NIntegrate in the range of accuracy from 20 to 80 decimal digits, (Romani, 1992).

Triple-Adaptive Quadrature

The efficiency of DAQ can be improved by exploiting the superior performance of the double exponential quadrature on some integrands with end-point singularities. In this section we given a sketch of the triple-adaptive algorithm (TAQ in the following) which attains the bounds

$N = O(E)$ for functions in class $\mathcal{A}[a, b, \delta]$ and
for functions in class $\mathcal{M}[a, b, s]$ with only apparent singularities;

$N = O(E \log E)$ for functions of class $\mathcal{ES}$;

$N = O(E^2)$ for the other functions in class $\mathcal{M}$.

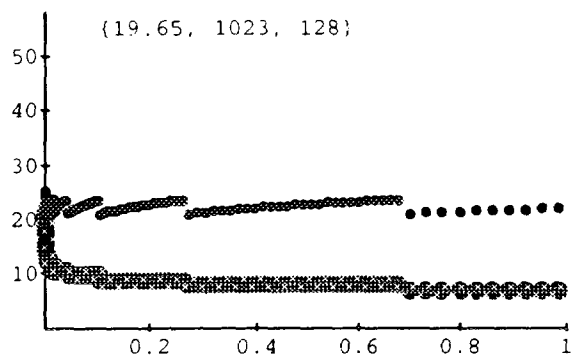

FIG. 5.3. Function $x^{3/2}$, Clenshaw–Curtis 7-point, 128 subintervals, Algorithm GQ. The meaning of symbols and axes is the same as Fig. 2.3.

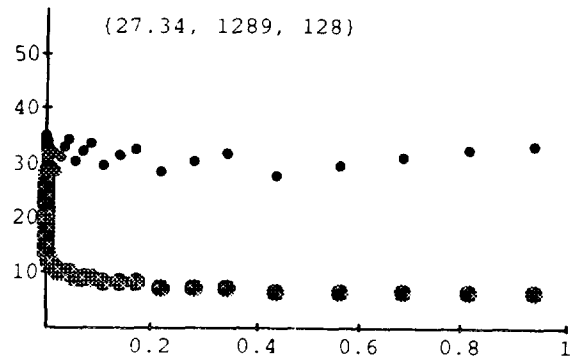

FIG. 5.4. Function $x^{3/2}$, 128 subintervals, Algorithm DAQ. The meaning of symbols and axes is the same as Fig. 2.3.

The basic idea consists of combining, in a general global adaptive schema, the three main strategies for improving the approximation of the integral, i.e., the interval subdivision, the application of increasing precision Clenshaw–Curtis quadrature, and the application of DE-quadrature. In other words, the active subintervals are ordered into a queue according to the error estimate, and at any step the subinterval with the worst error estimate can be bisected or processed with a higher degree formula. If an integration difficulty is detected in an interval containing an end-point then, before bisecting the interval, DE-quadrature is attempted. The quadrature program uses a local routine

```
procedure DErule(a, b: real; k: integer; var abserr, result: real)
```

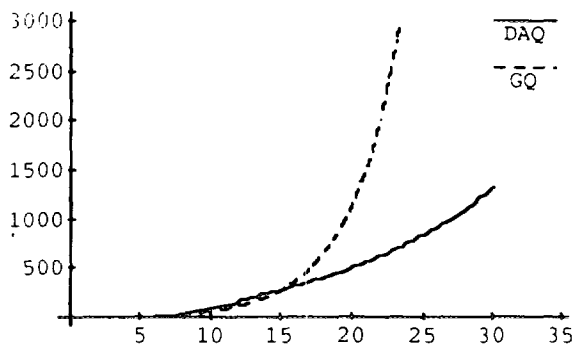

FIG. 5.5. N versus E for the programs GQ (using Clenshaw–Curtis 7-point rule) and DAQ, applied to the integrand $x^{3/2}$.

that implements DE-quadrature with step $h = 2^{-k}(b - a)$ choosing the smallest integration interval for which the estimated trimming error is negligible. The function `test(est, est1)` can be used to see whether the DE-quadrature converges slower than the theoretical expectations.

ALGORITHM 5.2.

```
procedure TAQuadrature (a0, b0, epquad: real; var abserr, result:
          real);
    var res,est,res1,est1,res2,est2,middle,a,b: real;
        k: integer;
{   k0: integer; is a global variable denoting the starting point
        for the local rule                                        }
{   procedure CCrule(a,b: real; k: integer; var abserr, result:
              real);                                              }
{             implements Clenshaw-Curtis Quadrature with 2^k+1
              points                                              }
{   procedure DErule(a,b: rule; k: integer; var abserr, result:
              real);                                              }
{             implements double exponential Quadrature with step
              h = 2^-k(b-a)                                       }
{   function test(est,est1 : real):boolean;                       }
{        tests the convergence after doubling the points in
         Clenshaw-Curtis quadrature                               }
{        or after halving h in DE quadrature                      }
{                                                                 }
{   put_interval_k and get_interval_k handle the priority queue,  }
{                                                                 }
{   k > 0 means that the kth Clenshaw-Curtis rule has been
applied                                                           }
{   k <-1 means that double exponential quadrature with step
          h = 2^-k(b-a)                                           }
          has been applied
{   k = 0 means that a singularity has been detected              }
{   k = -1 means that DE quadrature application was unsuccessful  }

begin
    CCrule(a0,b0,k0,abserr,result):                  { integrate on
                                                       the first
                                                       interval }
   put_interval_k(a0,b0,k0, abserr, result); { put the interval in
                                               the queue }
    while abserr > epquad do begin
         get_interval_k(a,b,k,est,res);           { get the interval
                                                    with the largest
                                                    error }
         if k>0 then begin {no singularity was detected for the
         interval [a,b]}
               k:=k+1;
               CCrule(a,b,k, est1, res1):
               result:=result-res+res1;          {update estimates}
```

```
            abserr:=abserr-est+est1;
            if test(est,est1) then k:=0; {test for convergence}
            put_interval_k(a,b,k,abserr,result)

        end else if k=0 then {a singularity was detected for the
                              interval [a,b]}
            if the subinterval does not contain an end-point
            then begin
                 middle:=(a+b)/2          {perform bisection}
                 CCrule(a,middle,k0,est1,res1);
                 put_interval_k(a,middle,k0, est1, res1);

                 CCrule(middle,b,k0,est2,res2);
                 put_interval_k(middle,b,k0,est2,res2);
                 result:=result-res+res1+res2;        {update
                                                       estimates}
                 abserr:=abserr-est+est1+est2

            end else begin
                DErule(a,b,2,est1,res1);    {perform DE quadrature}
                DErule(a,b,3,est2,res2);    {perform DE quadrature}
                if test(est1,est2)
                      then put_interval_k(a,b,-1, est, res);
                      else begin
                           put_interval_k(a,b,-3, est1, res1);
                           result:=result-res+res2;
                           abserr:=abserr-est+est2
                      end
                 end
            else if k=-1 then begin
                 middle:=(a+b)/2             {perform bisection}
                 CCrule(a,middle,k0,est1,res1);
                 put_interval_k(a,middle,k0, est1, res1);
                 CCrule(middle,b,k0,est2,res2);
                 put_interval_k(middle,b,k0,est2,res2);
                 result:=result-res+res1+res2;
                 abserr:=abserr-est+est1+est2

            end else{k<-1} begin
                DErule(a,b,-k+1,est1,res1); {perform DE quadra-
                                             ture}
                if test(est,est1)
                      then put_interval_k(a,b,-1, est, res);
                      else begin
                           put_interval_k(a,b,k-1, est1, res1);
                           result:=result-res+res1;
                           abserr:=abserr-est+est1
                      end
      end
    end;
end;
```

The computational cost of the algorithm can be discussed by investigating the following cases:

(1) the integrand is regular at the end-points. The DE-quadrature is not attempted and the behaviour is the same of DAQ.

(2) the integrand is singular at one or both the end-points and DE-quadrature achieves fast convergence. The integration in the intervals containing the singularities costs $O(E \log E)$ and the integration in the rest of the interval is done with cost $O(E)$;

(3) the integrand is singular at one or both the end-points and DE-quadrature achieves fast convergence in a neighbourhood of these singularities; moreover, a finite number of bounded singularities are located in the integration interval. The integration near the end-points costs $O(E \log E)$ and the integration in the rest of the interval is done with cost $O(E^2)$ by the same technique of DAQ.

(4) the integrand presents end-point integration difficulties which do not pass the test of fast convergence. The overall behavior is the same as DAQ with an additional cost due to the attempts made to check whether DE-quadrature is applicable. The asymptotic cost is of order $O(E^2)$.

6. Conclusion

The complexity results of the classical algorithms reviewed and of the two new ones inroduced here, are summarised in Table I. The quantities p and r have been defined previously. The bounds proved here or those for which the value of the constant has been derived in this paper, are shown in boldface.

TABLE I
Magnitude Order of the Number of Evaluations N vs the Number of Exact Decimal Digits in the Result

Algorithm	Class: $\mathcal{A}[a, b, \delta]$	$\mathcal{M}[a, b, s]$	$\mathcal{PP}$	$\mathcal{EF}$
Clenshaw–Curtis (CCQ)	E	—	—	—
Panel	$\mathbf{10^{E/p}}$	$10^{E/\min(r,s)}$, $s \geq 1$	—	—
Double exponential	$E \log E$	—	—	$E \log E$
Global adaptive (GQ)	$\mathbf{10^{E/r}}$	$\mathbf{10^{E/r}}$ [a]	E^c	$\mathbf{10^{E/r}}$ [a]
Double-adaptive (DAQ)	$\mathbf{E}$	$\mathbf{E^2}$ $\mathbf{E}^b$	$\mathbf{E}$	$\mathbf{E^2}$
Triple-adaptive (TAQ)	$\mathbf{E}$	$\mathbf{E^2}$ $\mathbf{E}^b$	$\mathbf{E}$	$\mathbf{E \log E}$

[a] This result has been proved under the additional hypothesis (2.1).
[b] If the integrand has only apparent singularities.
[c] If the polynomials have degree less than r.

APPENDIX

Error of Single Rule, Regular Functions

Consider an interpolatory formula $\mathcal{Q}$ with degree of precision $r - 1$ and a function $f \in C^s[a,b]$. Let I_{computed} be the result of the approximation of the integral I of f on $[a, b]$, obtained using $\mathcal{Q}$. The inequality

$$|I - I_{\text{computed}}| \leq \left(\frac{b-a}{2}\right)^{j+1} c_j \max_{x\in[a,b]} \frac{|f^{(j)}(x)|}{j!}, \qquad 1 \leq j \leq \min[r,s], \quad \text{(A.1)}$$

can be stated for the absolute value of the truncation error, where we denote by c_j the quantities:

$$c_j = j! \int_{-1}^{1} |K_{j-1}(t)|\, dt, \qquad 1 \leq j \leq r,$$

and $K_j(t)$ is the so-called Peano kernel defined as in Davis and Rabinowitz (1984, p. 286). For $\mathcal{A}[a, b, \delta]$ functions, the folloiwng important property of the derivatives of class $\mathcal{A}[a, b, \delta]$ functions is known (Davis and Rabinowitz, 1984, p. 301):

$$\max_{x\in[a,b]} \frac{|f^{(j)}(x)|}{j!} \leq \delta^{-j} \max_{z\in\partial\Gamma(a,b,\delta)} |f(z)|, \qquad j \geq 1. \quad \text{(A.2)}$$

Combining (A.1) and (A.2) we get the bound

$$|I - I_{\text{computed}}| \leq \left(\frac{b-a}{2}\right)^{r+1} c_r\, \delta^{-r} \max_{z\in\partial\Gamma(a,b,\delta)} |f(z)|. \quad \text{(A.3)}$$

An asymptotic representation of the error can be obtained by applying the results of Favati *et al.* (1992) which involve constants β_r and γ_{r+2}, related to the Peano coefficients, depending on $\mathcal{Q}$, only.

THEOREM A.1. *Let $\lambda > 0$ and let f be a $C^{r+2}[x - \lambda, x + \lambda]$ function. Let I_{computed} be the result of the approximation of the integral I of f on $[x - \lambda, x + \lambda]$, obtained using a symmetrical rule $\mathcal{Q}$. Then for the truncation error $I - I_{\text{computed}}$ the followng equality holds:*

$$I - I_{\text{computed}} = \lambda^{r+1} f^{(r)}(x) \frac{\beta_r}{r!} + \lambda^{r+3} \Delta,$$

with

$$|\Delta| \leq \frac{\gamma_{r+2}}{(r+2)!} \max_{u \in [x-\lambda, x+\lambda]} |f^{(r+2)}(u)|,$$

where β_r and γ_{r+2} are constants depending on $\mathfrak{Q}$, only.

Proof. The theorem is a direct consequence of Theorem 3.3 in Favati *et al.* (1992). ■

Theorem A.1 allows us to prove the asymptotic validity of Assumption 2.2. Consider two symmetrical rules $\mathfrak{Q}$ and $\mathfrak{Q}'$ of degrees of precision $r-1$ and $r'-1$, $r' > r$. Let $I_{Q'}$ and I_Q be the corresponding computed values of the interal. For $C^{r'+2}[x-\lambda, x+\lambda]$ functions we have

$$I - I_Q = \lambda^{r+1} \frac{\beta_r}{r!} f^{(r)}(x) + O(\lambda^{r+3})$$

and, analogously,

$$I - I_{Q'} = \lambda^{r'+1} \frac{\beta'_{r'}}{r'!} f^{(r')}(x) + O(\lambda^{r'+3}).$$

Since $r' > r$,

$$\begin{aligned} I_{Q'} - I_Q &= I - I_Q - \lambda^{r'+1} \frac{\beta_{r'}}{r'!} f^{(r')}(x) + O(\lambda^{r'+3}) \\ &= I - I_Q + O(\lambda^{r'+1}). \end{aligned}$$

This proves the asymptotic correctness of the trivial error estimate $I_{Q'} - I_Q$.

Error of Single Rule, Singular Functions

The following theorem provides an upper bound to the error that holds for all bounded integrands.

THEOREM A.2. *Let f be a bounded function in $[a, b]$. Suppose that f is integrated by an interpolatory rule with positive weights in an interval of length $[x-\lambda, x+\lambda] \subseteq [a, b]$. Then*

$$|I - I_{\text{computed}}| \leq B\lambda,$$

where B is a constant depending on f, a, b, and is independent of λ.

Proof. The proof follows by elementary calculus. ■

THEOREM A.3. *Let f be an unbounded function in $[0, 1]$ with only one singularity at 0. Let $\alpha < 1$, $|f(x)| = O(x^{-\alpha-\varepsilon})$, for any $0 < \varepsilon < 1 - \alpha$. Suppose that f is integrated on $[0, 2\lambda]$, $0 < \lambda \leq 1/2$, with an interpolatory rule Q with positive weights. Then*

$$|I - I_{\text{computed}}| \leq B\lambda^{(1-\alpha-\varepsilon)},$$

where B is a constant depending on ε, Q, f, and is independent of λ.

Proof. The proof follows by elementary calculus. ■

Error Behaviour after Bisections

THEOREM A.4. *Let $R = [x - \lambda, x + \lambda]$, $R_1 = [x - \lambda, x]$, $R_2 = [x, x + \lambda]$. Let f be a $C^{r+2}[x - \lambda, x + \lambda]$ function; let e, e_1, e_2, be the errors of the integrals of f on R, R_1, and R_2, respectively, computed by the symmetrical rule Q of degree $r - 1$. Then*

$$e_1 = 2^{-(r+1)}e + O(\lambda^{r+2}),$$
$$e_2 = 2^{-(r+1)}e + O(\lambda^{r+2}).$$

Proof. Applying Theorem A.1 to R, R_1, and R_2 the associated errors are

$$e = \lambda^{r+1}\frac{\beta r}{r!}f^{(r)}(x) + \lambda^{r+3}\Delta_0,$$

$$e_1 = 2^{-(r+1)}\lambda^{r+1}\frac{\beta r}{r!}f^{(r)}(x - \lambda/2) + 2^{-(r+3)}\lambda^{r+3}\Delta_1,$$

$$e_2 = 2^{-(r+1)}\lambda^{r+1}\frac{\beta r}{r!}f^{(r)}(x + \lambda/2) + 2^{-(r+3)}\lambda^{r+3}\Delta_2,$$

$$|\Delta_0|, |\Delta_1|, |\Delta_2| \leq \frac{\gamma_{r+2}}{(r+2)!}\max_{u\in R}|f^{(r+2)}(u)|.$$

Using the Taylor formula we can write

$$f^{(r)}(x \pm \lambda/2) = f^{(r)}(x) \pm \frac{\lambda}{2}f^{(r+1)}(x) + \frac{\lambda^2}{8}\Gamma, \qquad |\Gamma| \leq \max_{u\in[a,b]}|f^{(r+2)}(u)|.$$

Hence

$$e_1 = 2^{-(r+1)}e - 2^{-(r+2)}\lambda^{r+2}\frac{\beta r}{r!}f^{(r+1)}(x) + 2^{-(r+3)}\lambda^{r+3}\left(\frac{\beta r}{r!}\frac{\Gamma_1}{2} + \Delta_1 - 4\Delta_0\right)$$

$$= 2^{-(r+1)}e + O(\lambda^{r+2}),$$

$$e_2 = 2^{-(r+1)}e + 2^{-(r+2)}\lambda^{r+2}\frac{\beta r}{r!}f^{(r+1)}(x) + 2^{-(r+3)}\lambda^{r+3}\left(\frac{\beta r}{r!}\frac{\Gamma_2}{2} + \Delta_2 - 4\Delta_0\right)$$

$$= 2^{-(r+1)}e + O(\lambda^{r+2}), \qquad |\Gamma_1|, |\Gamma_2| \le \max_{u\in[a,b]}|f^{(r+1)}(u)|,$$

from which the result follows. ■

LEMMA A.1. *Let f be a function that is continuable analytically in the interior of $\Gamma(a + \delta, b, \delta)$; let*

$$\alpha = \inf\{\beta: |f(z)| = O(|z - a|^{-\beta}),\ z \in \Gamma(a + \delta, b, \delta)\} < 1;$$

and let $R = [x - \lambda, x + \lambda]$, $x > a$, $\lambda < (x - a)/2 < \delta$. Then, for any ε, $0 < \varepsilon < 1 - \alpha$, we have

$$\max_{u\in R}|f^{(j)}(u)| = O((x - a)^{-\alpha-\varepsilon-j}).$$

Proof. By using relation (A.2) on $\Gamma(x - \lambda, x + \lambda, (x - a)/4)$ we get

$$\max_{u\in R}\frac{|f^{(j)}(u)|}{j!} \le \left(\frac{x - a}{4}\right)^{-j}\max_{z_\varepsilon\,\partial\Gamma(x-\lambda,x+\lambda,(x-a)/4)}|f(z)| = O((x - a)^{-\alpha-\varepsilon-j}). \quad ■$$

THEOREM A.5. *Let f be a C^{r+2} function in $(y, b]$, with the properties:*

$$|f^{(r)}(x)| = O((x - y)^{-\alpha-\varepsilon-r}), \qquad \text{for any } \varepsilon > 0,\ x > y.$$

$$|f^{(r)}(x)| = \Omega((x - y)^{-\alpha+\varepsilon-r}), \qquad \text{for any } \varepsilon > 0,\ x > y.$$

Let $R = [x - \lambda, x + \lambda]$, $R_1 = [x - \lambda, x]$, $R_2 = [x, x + \lambda]$, $R \subset [a, b]$, $y \notin R$. Let, for some positive constant ε, $\lambda = o(|x - y|^{1+2\varepsilon})$, $x \to y$. Let e, e_1, e_2, be the errors of the integrals of f on R, R_1, and R_2, respectively, computed by the symmetrical rule Q of degree $r - 1$. Then we have

$$|e| \sim \lambda^{r+1}\frac{|\beta_r|}{r!}|f^{(r)}(x)|$$

$$e_1 = 2^{-(r+1)}e + o(e),$$

$$e_2 = 2^{-(r+1)}e + o(e).$$

Proof. Applying Theorem A.1 to R, R_1, and R_2, the associated errors can be written

$$e = \lambda^{r+1} \frac{\beta_r}{r!} f^{(r)}(x) + O(\lambda^{r+3}(x - y)^{-\alpha-\varepsilon-r-2}),$$

$$e_1 = 2^{-(r+1)}e + O(\lambda^{r+2}(x - y)^{-\alpha-\varepsilon-r-1}) + O(\lambda^{r+3}(x - y)^{-\alpha-\varepsilon-r-2}),$$

$$e_2 = 2^{-(r+1)}e + O(\lambda^{r+2}(x - y)^{-\alpha-\varepsilon-r-1}) + O(\lambda^{r+3}(x - y)^{-\alpha-\varepsilon-r-2}).$$

We have

$$\lambda^{r+1}|f^{(r)}(x)| = \Omega(\lambda^{r+1}(x - y)^{-\alpha-r+\varepsilon}),$$

since

$$\lim_{x\to y} \frac{\lambda^{r+3}(x - y)^{-\alpha-\varepsilon-r-2}}{\lambda^{r+1}(x - y)^{-\alpha-r+\varepsilon}} = \lim_{x\to y} \left(\frac{\lambda}{(x - y)^{1+\varepsilon}}\right)^2 = 0,$$

and thus it follows that

$$|e| \sim \lambda^{r+1} \frac{|\beta_r|}{r!} |f^{(r)}(x)| = \Omega(\lambda^{r+1}(x - y)^{-\alpha-r+\varepsilon}).$$

Analogously

$$\lim_{x\to y} \frac{\lambda^{r+2}(x - y)^{-\alpha-\varepsilon-r-1}}{\lambda^{r+1}(x - y)^{-\alpha-r+\varepsilon}} = \lim_{x\to y} \frac{\lambda}{(x - y)^{1+2\varepsilon}} = 0,$$

and the result follows. ■

The same result can be trivially derived for the interval $[a, y)$.

LEMMA A.2. *Let $\varphi(x)$, $\eta(x)$ be nonnegative functions and let $r > 1$; then*

$$\left(\int_a^b \varphi(x)/\eta(x)\, dx\right)\left(\int_a^b \varphi(x)\eta(x)^r dx\right)^{1/r} \geq \left(\int_a^b \varphi(x)dx\right)^{(r+1)/r}, \quad \text{(A.4)}$$

where the equality holds if and only if $\eta(x)$ is a constant.

Proof. Let us write, for two nonnegative functions $f(x)$ and $g(x)$, the Holder inequality,

$$\left(\int_a^b f(x)^p dx\right)^{1/p}\left(\int_a^b g(x)^q dx\right)^{1/q} \geq \int_a^b f(x)g(x)\, dx, \qquad p, q > 1, \frac{1}{p} + \frac{1}{q} = 1,$$

and the equality holds if and only if $f(x)^{p-1}/g(x)$ is a constant. Raising both sides of (A.4) to the $r/(r+1)$ power we get

$$\left(\int_a^b \varphi(x)/\eta(x)\,dx\right)^{r/(r+1)}\left(\int_a^b \varphi(x)\eta(x)^r dx\right)^{1/(r+1)} \geq \int_a^b \varphi(x)dx.$$

Let $p = (r+1)/r$ and $q = r+1$ and let $f(x) = [\varphi(x)/\eta(x)]^{r/(r+1)}$ and $g(x) = [\varphi(x)\eta(x)^r]r/(r+1)$; then $f(x)g(x) = \varphi(x)$ and $f(x)^{p-1}/g(x) = \eta(x)^{-1}$ and the result follows. ∎

LEMMA A.3. *Consider an interval* $[y, b]$ *and a point* $t \in (y, b]$. *For a fixed constant* $\varepsilon > 0$, *an interval* $[x-\lambda, x+\lambda] \subset [t, b]$ *is called "large" if* $\lambda \geq (|x-y|+\lambda)^{1+3\varepsilon}$, *and "small" otherwise. Then we have*

(1) *the number of large intervals in* $[t, b]$ *is of order* $O(|t-y|^{-3\varepsilon})$ *for* $t \to y$.

(2) *for small intervals* $\lambda = o(|x-y|^{1+2\varepsilon})$ *for* $x \to y$.

Proof. (1) Let us split the interval $[t, b]$ into k subintervals $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, k-1$, $x_0 = t$, $x_k = b$, and let m be the number of large subintervals. Let $2\lambda(u)$ denote the length of the subinterval containing the point $u \in [t, b]$; the density of subintervals is $(2\lambda(u))^{-1}$ and, for large subintervals, we have $\lambda(u) \geq |u-y|^{1+3\varepsilon}$. Let L be the union of large subintervals; then

$$m = \frac{1}{2}\int_L \lambda(u)^{-1}du \leq \frac{1}{2}\int_L |u-y|^{-1-3\varepsilon}du \leq \frac{1}{2}\int_t^b |u-y|^{-1-3\varepsilon}du$$
$$= O(|t-y|^{-3\varepsilon}).$$

(2) Let $s = (x-y)$, then we have for small intervals $\lambda < (s+\lambda)^{1+3\varepsilon}$; moreover, $\lambda \to 0$ for $s \to 0$ and

$$s > s + \lambda - (s+\lambda)^{1+3\varepsilon} = (s+\lambda)(1-(s+\lambda)^{3\varepsilon}).$$

Then

$$\frac{\lambda}{s^{1+2\varepsilon}} < \frac{(s+\lambda)^{1+3\varepsilon}}{s^{1+2\varepsilon}} < \frac{s^{1+3\varepsilon}}{s^{1+2\varepsilon}(1-(s+\lambda)^{3\varepsilon})^{1+3\varepsilon}}$$

and

$$\lim_{s\to 0}\frac{\lambda}{s^{1+2\varepsilon}} = 0. \quad \blacksquare$$

The same result can be analogously derived for the interval $[a, y)$.

REFERENCES

CHAWLA, M. M. (1968), Error estimates for Clenshaw–Curtis quadrature, *Math. Comp.* **22,** 651–656.

CLENSHAW, C. W., AND CURTIS, A. R. (1960), A method for numerical integration on an automatic computer, *Numer. Math.* **2,** 197–205.

DAVIS, P. J., AND RABINOWTIZ, P. (1984), "Methods of Numerical Integration," Academic Press, New York.

DEBOOR, C., AND RICE, J. R. (1979), An adaptive algorithm for multivariate approximation giving optimal convergence rates, *J. Approx. Theory* **25,** 337–359.

FAVATI, P., LOTTI, G., AND ROMANI, F. (1991a), Interpolatory integration formulas for optimal composition, *ACM Trans. Math. Software* **17,** 207–217.

FAVATI, P., LOTTI, G., AND ROMANI, F. (1991b), Improving QUADPACK automatic integration routines, *ACM Trans. Math. Software* **17,** 218–232.

FAVATI, P., LOTTI, G., AND ROMANI, F. (1992), Asymptotic expansion of error in interpolatory quadrature, *Comput. Math. Appl.* **34,** 99–104.

GENTLEMAN, G. M. (1972), Implementing the Clenshaw–Curtis quadrature, *Comm. ACM* **15,** 337–360.

GUI, W., AND BABUSKA, I. (1986). The h, p, and $h - p$ versions of the finite element method in 1 dimension. Part III. The adaptive $h - p$ version, *Numer. Math.* **49,** 659–683.

KAHANER, D., AND STOER, J. (1983), Extrapolated adaptive quadrature, *Siam. J. Sci. Stat. Comput.* **4,** 31–44.

MALCOLM, M. A., AND SIMPSON, R. B. (1975), Local versus global strategies for adaptive quadrature, *ACM Trans. Math. Software* **1,** 129–146.

MUROTA, K., AND IRI, M. (1982). Parameter tuning and repeated application of the IMT-type transformation in numerical quadrature, *Numer. Math.* **38,** 347–363.

OLIVER, J. (1972). A doubly-adaptive Clenshaw–Curtis quadrature method, *Comput. J.* **15,** 141–147.

PIESSENS, R., DEDONCKER-KAPENGA, E., ÜBERHUBER, C., AND KAHANER, D. K., (1983), "Quadpack: A Subroutine Package for Automatic Integration," Springer-Verlag, Berlin.

RICE, J. R. (1975), A metalgorithm for adaptive quadrature, *J. Assoc. Comput. Mach.* **22,** 61–82.

RIESS, R. D., AND JOHNSON, L. W. (1972), Error estimates for Clenshaw–Curtis quadrature, *Numer. Math.* **18,** 345–353.

ROMANI, F. (1992), High precision automatic quadrature, *in* "European Mathematica Conference, Rotterdam."

STENGER, F. (1978), Optimal convergence of minimum norm approximations in H_p, *Numer. Math.* **29,** 345–362.

TAKAHASI, H., AND MORI, M (1974), Double exponential formulas for numerical integration, *Publ. Res. Inst. Math. Sci,* **9,** 721–741.

TRAUB, J. F., WASILKOWSKY, G. W., AND WOZNIAKOWSKY, H. (1988), "Information-Based Complexity," Academic Press, New York.