THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 9, November 1977
Printed in U.S.A.

Automatic Numerical Quadrature

By JAMES L. BLUE

(Manuscript received April 15, 1977)

An automatic numerical quadrature routine (ANQR) attempts to evaluate

$$\int_a^b f(x)\,dx$$

to absolute accuracy ϵ, given only ϵ, a, b, and a user-supplied subroutine which calculates f(x) for any x in [a,b]. An ANQR which guarantees success is impossible to construct, even disregarding the effects of finite computer precision, but the problem is nonetheless of interest. A reliable and efficient ANQR is a necessary part of any mathematical subroutine library. New single- and double-precision ANQRs, QUAD and DQUAD, have been constructed and tested. They are based on adaptive Romberg extrapolation, with cautious error estimation. An important practical feature is the automatic recognition of endpoint singularities, and a change of variable to handle them. QUAD and DQUAD also recognize the presence of noise in the function being integrated, and limit the attempted accuracy accordingly. Since guaranteed ANQRs are impossible, extensive testing of DQUAD is presented to demonstrate its efficiency and robustness. Comparable testing is not available for competitive ANQRs, but performance on a standard set of test integrals is presented for DQUAD and nine other ANQRs. DQUAD is generally better. QUAD and DQUAD are written in PFORT, a subset of American National Standard (ANS) Fortran. Machine-dependent constants are obtained from the PORT library machine-constants programs. A portable package of storage allocation routines is used.

I. INTRODUCTION

The development of automatic numerical quadrature routines (ANQRs) has been a popular research topic for many years (see refs. 1–6, 8, 9, 11, 13, 14). An ANQR is a routine which attempts to calculate

$$\int_a^b f(x)\,dx$$

with absolute error, or perhaps relative error, no larger than ϵ, given ϵ, a, b, and a procedure which calculates $f(x)$ for any desired x in the interval $[a,b]$. It is assumed that no other information about the function f is available. The problem is perhaps the more interesting for being an impossible one. Any numerical quadrature routine must estimate the integral by sampling the function f at a finite number of x's. A guaranteed automatic integration algorithm is clearly impossible for general f, even for analytic f. For example, given any deterministic rule for numerical quadrature, one can readily find constants α and β so that the quadrature rule calculates

$$\sqrt{\alpha} \int_0^1 e^{-\alpha(x-\beta)^2}\, dx$$

to be close to zero. (Choose α to be large and positive, and β to be between sampling points.)

Although the general problem is impossible, one feels that an ANQR which works for "reasonable" functions should be feasible, and much work has been directed at this goal. There has been great confusion and difficulty in comparing the various candidates for ANQRs, partly because the domain of the problem is undefined; a reasonable definition of a "reasonable" function is itself difficult.

In constructing an ANQR, an author is forced to make decisions about the class of "reasonable" functions, in effect to define what is a "reasonable" function. These decisions strongly affect the efficiency and robustness of the ANQR. For example, to avoid completely missing an isolated peak in $f(x)$, the interval $[a,b]$ must be sampled finely. However, a fine sampling is inefficient for easy functions. Another example is a function which is flat over 99 percent of the interval, and which has 200 oscillations in the remaining 1 percent. If an ANQR is able to distinguish this function from one which is merely noisy over 1 percent of the interval, the ANQR is likely to be inefficient on easy functions and very inefficient on noisy functions. An ANQR which gives up relatively quickly on this function, calls it noisy, and returns an error message, may be preferable, especially since many such functions are the result of a user's programing errors.

A compromise strategy, used by QUAD, is to isolate all assumptions about the "reasonable" class of functions in a few parameters. Default values of these parameters can be chosen which will be suitable for most users. More knowledgeable users can use other values. With the default values, QUAD strikes what the author considers to be the proper balance between efficiency and robustness.

Since no *a priori* information about $f(x)$ is available,

$$\int_a^b f(x)\, dx$$

must be evaluated by sampling f in $[a,b]$; the error in the calculated integral is usually estimated by comparing two or more calculated values for the integral. ANQRs typically have a sequence of quadrature rules Q_n, depending on a, b, and the function f, such that

$$\lim_{n\to\infty} Q_n = \int_a^b f(x)\,dx$$

if the calculations are done in infinite precision, and if f is at least piecewise continuous. Most ANQRs have no better error estimation procedure than to accept Q_n whenever $|Q_{n-1} - Q_n| < \epsilon$, a procedure fraught with danger. QUAD has a much more stringent error estimation procedure, described in Section II.

Many functions to be integrated are easy to integrate over some parts of the interval and difficult over other parts. It is frequently more efficient to sample more densely in the difficult regions, if possible. ANQRs which attempt to do this are called *adaptive*—the points at which f is sampled depend on the function being sampled. An adaptive ANQR must include some strategy for how to concentrate the sampling points. Essentially all competitive ANQRs are adaptive.

The usual adaptive procedure is to integrate an interval with quadrature rules Q_n for $n = 1, 2, \ldots, N$, where N is fixed. Q_n may be, for example, Simpson's rule with 2^N intervals, or Gauss-Legendre quadrature with n sampling points. If convergence has not been obtained, the interval is divided in half, and each half considered separately. For efficiency, one wants quadrature rules for which all sampling points for the whole interval are also used for the half-intervals. If the value of N used depends on the results Q_n for $n < N$, the method is sometimes called *doubly-adaptive*.

Most ANQRs do not do well on integrals with endpoint singularities, but users' integrals are frequently of this type. QUAD has a provision for recognizing endpoint singularities and for making a change of variable to facilitate the integration. This feature also works well on another important class of functions, those decaying steeply away from one or both ends of the integration interval. This automatic change of variable technique is a significant improvement over previous ANQRs.

Most ANQRs cannot cope with noisy functions; if there is too much noise in f, most ANQRs fail in an unpleasant, uneconomical way. Convergence will be at best very slow, so that the ANQRs will stop only when their predefined limit on calls to the function evaluation procedure has been exceeded, with no indication that the problem is noise rather than a noise-free but unruly function f. QUAD recognizes noisy functions, sets a warning flag, and integrates only to an accuracy commensurate with the estimated noise.

Finally, there is a large difference between an algorithm for numerical

quadrature and a properly-written ANQR suitable for a program library. Provision must be made, for example, to stop trying to integrate a function if it has been sampled more than some user-defined number of times. The finite machine precision of the computer involved must be taken into account. Temporary storage must not be allowed to overflow. Provision for error returns must be made.

The basic idea behind QUAD is adaptive Romberg extrapolation[5] combined with cautious error estimation[9]. The first such combination was the program CADRE, written by deBoor[6]. CADRE and QUAD are superficially similar, but differ in almost every detail. The major improvements incorporated in QUAD include the following, which will be covered fully in Section II.

(*i*) Noise. QUAD detects noisy functions and quits gracefully.

(*ii*) Endpoint singularities. QUAD detects singularities in $f(x)$ at the endpoints, a and b, and automatically makes a change of variable to reduce the strength of the singularity.

(*iii*) Mesh sequence. QUAD uses the mesh sequence 1, 2, 3, 4, 6, 8, 12, 16, . . . , instead of 1, 2, 4, 8, 16, . . . , giving a higher effective order of convergence.

(*iv*) Portability. QUAD is written in PFORT,[15] a portable subset of ANS Fortran. Machine-dependent quantities are defined with the PORT[7] machine constants. A portable Fortran stack[7] is used for temporary storage.

Section II discusses the algorithm of QUAD and DQUAD more fully. Section III compares the performance of DQUAD and nine competitive ANQRs on a standard set of test integrals, and also presents the results of some more serious testing of QUAD. Section IV discusses the implementation of QUAD, including portability considerations.

II. QUAD

2.1 Romberg extrapolation

QUAD is based on Romberg extrapolation of the composite trapezoidal rule.[5] The formulas are standard, but will be repeated here for completeness. Let $n_1, n_2, \ldots$ be an increasing sequence of positive integers, and let $h_i = (b - a)/n_i$. Then the composite trapezoidal approximation to

$$I = \int_a^b f(x)\,dx$$

is

$$T(h_i) = \frac{1}{2} h_i [f(a) + f(b)] + h_i \sum_{m=1}^{n_i - 1} f(a + mh_i)$$

If f has $2k + 1$ continuous derivatives in $[a,b]$, then the Euler-Maclaurin sum formula shows that

$$T(h_i) = I + \sum_{m=1}^{k} c_m h_i^{2m} + O(h_i^{2k+1})$$

where the c_m depend only on a, b, and f, not on h_i. A higher-order, although not necessarily more accurate, estimate may be obtained by combining two trapezoidal estimates via Richardson extrapolation,[3] eliminating the $c_1 h_i^2$ term. Let $T_0^{(i)} = T(h_i)$.

$$T_1^{(1)} = T_0^{(2)} + \frac{T_0^{(2)} - T_0^{(1)}}{h_1^2/h_2^2 - 1}$$
$$= I + O(h_1^2 h_2^2)$$

Still higher-order estimates may be generated recursively. The general formula for generating $T_k^{(i)}$ is

$$T_k^{(i)} = T_{k-1}^{(i+1)} + \frac{T_{k-1}^{(i+1)} - T_{k-1}^{(i)}}{h_i^2/h_{i+k}^2 - 1}$$

It is customary to think of the T-values as a table, *viz.*

$$\begin{array}{ccccc} T_0^{(1)} & & & & \\ T_0^{(2)} & T_1^{(1)} & & & \\ T_0^{(3)} & T_1^{(2)} & T_2^{(1)} & & \\ T_0^{(4)} & T_1^{(3)} & T_2^{(2)} & T_3^{(1)} & \\ T_0^{(5)} & T_1^{(4)} & T_2^{(3)} & T_3^{(2)} & T_4^{(1)} \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \end{array}$$

The classical Romberg method uses the sequence 1, 2, 4, 8, . . . for the n_i's.

Since at any time the right-most element, the "tip," of the T table is of highest order, it is expected to be most accurate, and frequently is so. Early Romberg programs[1] tested for convergence only by checking successive tip elements of the table. There are several arguments against this practice. Firstly, for the tip there is no way to obtain an error estimate with any theoretical foundation. Secondly, highest-order is not the same as most accurate. Even for analytic functions, if the step size used at the beginning of the T table is too large, the tip of the table may not be the most accurate value. Thirdly, for functions $f(x)$ which do not have enough derivatives, the tip of the T table is not of higher order than the elements to the left. For example, if $f(x) = x^\alpha$, for $0 < \alpha < 1$, the lowest-order term in each column of the kth row is $O(h^{\alpha+1})$. In practice it is frequently found that lower-order columns are more accurate than the tip element.

2.2 Cautious error estimation

The idea of cautious error estimation comes from Lynch.[9] It is a simple and seemingly unobjectionable idea, but is not adopted by most authors of ANQRs. A cautious error estimation procedure believes an error estimate only if there is some evidence that the convergence rate of successive quadrature rules is close to the theoretical rate. Cautious error estimation is particularly easy for Romberg extrapolation. The two were first combined by deBoor.[6] QUAD's version of catious error estimation is similar in spirit to deBoor's, but is more cautious and is different in all details.

It has been proven[1] that, if f is merely Riemann-integrable, each column of the T table converges, as does each diagonal. If f has enough continuous derivatives,

$$T_k^{(i)} = I + O(h_i^2 \ldots h_{i+k}^2)$$

These theoretical results provide a basis for cautious error estimation.[6,9] Lynch's suggestion was to consider three successive trapezoidal rule estimates, and form the ratios

$$R_0^{(i)} = \frac{T_0^{(i)} - T_0^{(i+1)}}{T_0^{(i+1)} - T_0^{(i+2)}}$$

$$= \frac{h_i^2 - h_{i+1}^2 + O[(h_i^2 + h_{i+1}^2)^2]}{h_{i+1}^2 - h_{i+2}^2 + O[(h_{i+1}^2 + h_{i+2}^2)^2]}$$

If the step sizes are small enough, the higher-order terms are small compared to the second-order terms, and

$$R_0^{(i)} \approx \frac{h_i^2 - h_{i+1}^2}{h_{i+1}^2 - h_{i+2}^2}$$

The calculated $R_0^{(i)}$ being close to this theoretical value is good evidence that the convergence rate of the column is proper, and that the error in $T_0^{(i)}$ is dominated by $c_1 h_i^2$. Then the Runge estimate of the error,[3]

$$|T_0^{(i+2)} - I| \approx \left| \frac{T_0^{(i+2)} - T_0^{(i+1)}}{h_{i+1}^2/h_{i+2}^2 - 1} \right| = |T_0^{(i+2)} - T_1^{(i+1)}|$$

is likely to be a good estimate. If the calculated $R_0^{(i)}$ is not close to the theoretical value, the Runge estimate of the error is likely to be an underestimate.

Similar calculations are done in higher columns. The general formula for $R_k^{(i)}$ is

$$R_k^{(i)} = \frac{T_k^{(i)} - T_k^{(i+1)}}{T_k^{(i+1)} - T_k^{(i+2)}}$$

As all the h's approach zero,

$$R_k^{(i)} \approx \frac{h_{i+1}^2}{h_{i+k+1}^2} \frac{h_i^2 - h_{i+k+1}^2}{h_{i+1}^2 - h_{i+k+2}^2}$$

The Runge estimate of the error in $T_k^{(i+2)}$ is

$$|T_k^{(i+2)} - I| \approx \left| \frac{T_k^{(i+2)} - T_k^{(i+1)}}{{h_i}^2/{h_{i+k+1}}^2 - 1} \right|$$

A more conservative estimate of the error in $T_k^{(i+2)}$,

$$|T_k^{(i+2)} - I| \approx |T_k^{(i+2)} - T_k^{(i+1)}|$$

is often used.

QUAD calls column k *asymptotic* if $R_k^{(i)}$ is close enough to the theoretical value; the tolerance is 5 percent of the theoretical value for $k = 0$, 10 percent for $k = 1$, 15 percent for $k = 2$, and so on. Column k is *almost asymptotic* if $R_k^{(i)}$ is between 0.25 and 4.0 times the theoretical value, except for column 0, where the criteria are 0.75 and 1.25.

If columns 0 through k are asymptotic, QUAD believes the Runge estimate for the error in the kth column. If columns 0 through $k - 1$ are asymptotic and column k is almost asymptotic, QUAD believes the conservative estimate for the error in the kth column, but no higher columns are believed. (The only exception is that, if the column 0 is only almost asymptotic, the next column is believed if it itself is asymptotic.)

This describes the basic cautious error estimation procedure for QUAD. There are a few more details, however. QUAD does not believe any answer based on less than two extrapolations, or five sampling points, per interval. If two successive entries in a column give the same value to within a few rounding errors, as occurs when integrating a constant function or in doing very accurate integration, then the column does not appear to be asymptotic. The conservative error estimate for the column is believed anyway. If an interval has a singularity, either real or due to rounding errors or truncation errors in the function subprogram for $f(x)$, no column will appear asymptotic. Then a nonasymptotic answer in the first column will be accepted after several extrapolations, with the very conservative error estimate

$$|T_0^{(i+2)} - I| \approx 2|T_0^{(i+2)} - T_0^{(i+1)}| + 2|T_0^{(i+1)} - T_0^{(i)}|$$

Finally, at any stage one column of the T table has only two entries in it, and cannot be judged to be asymptotic or nonasymptotic. The conservative error estimate is accepted for such a column if the previous column is asymptotic.

2.3 Step size sequence

The above discussion applies to any sequence of step sizes. The classical Romberg sequence uses step sizes $(b - a)/n$, with $n = 1, 2, 4, 8, 16, \ldots$, halving the step size and doubling the number of sampling points at each new extrapolation. Several alternative sequences have been suggested which do not cause the number of sampling points to rise so rapidly. QUAD uses n's of 1, 2, 3, 4, 6, 8, 12, . . .; another reasonable possibility is 1, 2, 3, 4, 5, 6, 8, 10, 12, ... These sequences double the number of sampling points every second and third extrapolation, respectively. The classical sequence has the advantages that the bookkeeping is very easy and that all old sampling points are reused if the interval is divided in half. The latter is essential for efficiency.

QUAD's sequence uses fewer sampling points to get the same accuracy, as suggested by Bulirsch and Stoer.[3] The bookkeeping is more complicated than for the classical sequence. For example, it is only convenient to divide the interval in half after the fourth, sixth, eight, . . . extrapolations if all the old sampling points are to be reused.

2.4 Adaptive procedure

For an adaptive Romberg extrapolation routine, it is necessary to decide when to do another trapezoidal rule and another extrapolation, and when to divide the interval. The minimum number of extrapolations for QUAD is 4, using step sizes $(b - a)$ down through $(b - a)/6$, and a total of 9 sampling points. This default lower limit may be raised by the user (see Section IV). Because of roundoff, unlimited extrapolations are impractical—the highest-order columns will not be asymptotic and will not be believed. The maximum number of extrapolations allowed by QUAD is 6; DQUAD allows 8. The default limit may be changed by the user (see Section IV).

If the requested error tolerance for an interval has not been achieved after 4 extrapolations, QUAD goes on to 5 and 6 extrapolations if the first column is asymptotic; after 6, it goes on if the second column is also asymptotic. This procedure is biased in favor of doing more extrapolations, and trying to get higher-order convergence, for smooth functions. Functions which are not smooth, or which do not appear asymptotic because of too large a step size, have the interval divided instead of having more extrapolations done. If QUAD decides to divide an interval, the lower half is stacked, and the upper half is attempted next.

2.5 Change of variable

Functions with singularities are expensive to integrate without special methods. Since the interval containing the singularity will have a nonasymptotic T table, convergence will be limited to the first column.

For example, it can be shown[10] that for

$$I = \int_0^1 f(x)\,dx = \int_0^1 x^{\alpha} g(x)\,dx \tag{1}$$

where g is smooth, and $f(0)$ is set equal to zero,

$$T_1(h) = I + \sum_{m=1}^{\infty} c_m h^{2m} + \sum_{m=1}^{\infty} d_m h^{m+\alpha} \tag{2}$$

The dominant error term for the first column of the T table is likely to be the $d_1 h^{1+\alpha}$ term, so convergence is slow. If $f(0)$ is not zero, another infinite sum is added to (2), like the second sum, but with $\alpha = 0$.

For such an endpoint singularity, the error is of a simple form. It is feasible to recognize this type of singularity in the same way that the cautious error estimation procedure recognizes asymptotic, or h^{2m} behavior. De Boor[6] does exactly this, estimates an $\bar{\alpha}$, and then extrapolates using eq. (2). The success of this procedure depends critically on how accurately α can be estimated. If $f(0)$ is not set equal to zero, de Boor's method will not work well. For logarithmic singularities, the error expansion corresponding to eq. (2) is more complicated; de Boor makes no attempt to recognize logarithmic singularities.

After recognizing an endpoint singularity, QUAD uses a different procedure. Suppose that the integral is as above, where g is well-behaved. Then the leading error term is $O(h^{1+\alpha})$ if $-1 < \alpha < 1$. In the second and higher columns, the $O(h^2)$ term is gone, so the leading term is $O(h^{1+\alpha})$ for $-1 < \alpha < 3$. QUAD looks at ratios of T table entries in the second and third columns to recognize $x^{\bar{\alpha}}$ behavior, and estimates the value of $\bar{\alpha}$. QUAD then makes a change of variable $x = u^n$, where n is the closest integer to $6/(1 + \bar{\alpha})$, giving for eq. (1)

$$\int_0^1 n u^{n-1} f(u^n)\,du = \int_0^1 n u^{n(1+\alpha)-1} g(u^n)\,du$$

(The change of variable is somewhat more complicated if the limits of integration are not 0 and 1.) The new integral has a singularity of the form u^{β}, where β is between 4.5 and 5.5, if $\bar{\alpha}$ is close to the true α. The singularity in the transformed integral is lessened, allowing convergence in the second or third columns of the T table. Convergence is likely to be much quicker. For rapid convergence, the method does not rely on the estimated $\bar{\alpha}$ being close to the true α or upon eq. (2) holding, or indeed on there being any singularity at all at the endpoint.

Steeply decaying integrands such as

$$100 \int_0^1 e^{-100x}\,dx$$

look like step functions when coarsely sampled. A step function is an x^0

singularity, since $f(0) = 0$, so a change of variable is made with $n = 6$. Singularities of the form $x^\beta \log(x)$ will look sufficiently like x^α singularities for some α, so that the transformation will be made. (It is not obvious that this is true, but tests have strongly indicated that it is.)

If $\bar{\alpha}$ is close to -1, n can become large. QUAD requires n to be less than a maximum value determined by the precision of the computer; this value was 22 for tests reported in Section III. The change of variable is not made if $\bar{\alpha}$ is less than -0.99.

To facilitate the change of variable, QUAD starts by dividing the interval $[a,b]$ into three equal intervals, and reverses the upper third. (Three is the default number, and may be changed by the user—see Section IV.) On the lower third and the reversed upper third, a left-hand endpoint singularity is recognized by a pattern of "fail on whole interval, succeed on right half-interval" twice in a row, and is followed by the estimation of $\bar{\alpha}$. If the two estimated values for $\bar{\alpha}$ from the second and third columns of the T table do not agree to within 0.1, no change of variable is made. No change of variable is attempted except at the two endpoints of the originial interval.

2.6 Noisy functions

All procedures for evaluating $f(x)$ are inherently noisy, since they are implemented on finite-precision machines. The value returned is not the exact $f(x)$, but $f(x)\,[1 + r_1(x)] + r_2(x)$, where $r_1(x)$ and $r_2(x)$ are noise functions. Ideally, r_1 could always be no larger than a few rounding errors, and r_2 could be no larger than a few times the smallest positive machine-representable number. Noise of this size should not affect the performance on an ANQR unless ϵ is very small, of the order of $r_2(x)$ or $f(x)r_1(x)$.

Protecting against this magnitude of noise is quite easy, although few ANQRs bother to do so. QUAD estimates *a priori* the sizes of r_1 and r_2, based on the machine precision, and requires all error tolerances to be at least as large as the estimated rounding error.

Protecting against significantly larger noise is more difficult. A successful ANQR should recognize the presence of noise, estimate its magnitude, and evaluate the integral in a "reasonable" number of function evaluations with an accuracy which is "nearly" as good as possible. (There is of course a trade-off between "reasonable" and "nearly.") If typical values of $r_2(x)$ or of $f(x)r_1(x)$ are much larger than $\epsilon/|b - a|$, most ANQRs will fail in an unpleasant, uneconomical way. Convergence will at best be very slow, so that the ANQRs will stop only when their predefined limit on calls to the function evaluation procedure has been exceeded, with no indication that the problem is noise rather than a noise-free but unruly function f.

SQUANK[11] makes a reasonable effort at recognizing noise. However, it attributes any nonstandard behavior to noise, so that some unruly but noise-free functions are called noisy.

Qualitatively, one may say that a function is noisy if, on a "sufficiently small" interval, the values of the samples of f are "not smooth enough." An algorithm consists of the defining of "sufficiently small" and "not smooth enough," followed by estimation of the magnitude of the noise and by further action to avoid using an excessive number of function values.

In QUAD, no answer is believed unless the function has been sampled with adjacent samples no farther apart than $h_s|b-a|$; h_s is supposedly small enough so that all structure may be seen by sampling with this spacing (h_s is parameter HSAMPL of Section IV). The default value of h_s is $1/8$. Noise is not estimated unless adjacent samples are no farther apart than $h_n|b-a|$, with $h_n = h_s/32$. Choosing h_n smaller would require more function values; choosing h_n larger would increase the risk of calling a function noisy when it is noise-free but rapidly varying.

When a function is noisy, QUAD will usually fail on large intervals, and then attempt smaller and smaller subintervals. When integration on a subinterval has failed, and adjacent samples are no farther apart than $h_n|b-a|$, the noise in f is estimated. First, the second differences of the samples of f on the subinterval are formed, e.g. $f(x) - 2f(x+h) + f(x+2h)$. If the sequence of second differences has no more than two sign changes, noise is not assumed to be present. If there are three or more sign changes, noise is assumed to be present, since the function has too much structure over too small an interval, and the estimated answer and error for that subinterval are accepted as being as good as possible.

When noise has thus been found to be present, the second differences are assumed to be essentially all noise, and the magnitude of the noise is estimated as the average of the absolute values of the second differences. All succeeding subintervals are attempted with accuracy not exceeding the estimated noise magnitude times the length of the subinterval. If other subintervals are found to be noisy, the largest noise magnitude is used.

2.7 Error allocation

QUAD attempts to integrate the upper one-third of $[a,b]$ with error tolerance $\epsilon/3$. Then the following procedure is used to assign an attempted accuracy for each interval. When integration on an interval is attempted with error tolerance ϵ_0 and fails, the upper half is attempted with error tolerance $\epsilon_0/2$. When integration on an interval succeeds, the absolute value of the estimated error is added to a running sum, and the top interval is the stack is attempted. If the top interval is of length δx, the total length of intervals remaining to be integrated is x_1, and the sum

of the absolute values of the error estimates so far is ϵ_1, then the error tolerance assigned to the top interval in the stack is $(\epsilon - \epsilon_1)\ \delta x/x_1$. However, the error assigned to any interval is required to be at least as large as $\epsilon/1000$.

If an answer is returned for an interval with an error estimate which is larger than the requested error, but less than the estimated roundoff or noise, that answer and error are accepted, and a warning flag is set.

III. TESTING AND COMPARISON OF ROUTINES

Testing is necessary to evaluate the efficiency and robustness of an ANQR. Typically the proposer of an ANQR generates an algorithm, codes a simple program, and tests the routine on a few integrals of the proposer's own choosing. It is not unusual for all the test integrals to be done well by the ANQR. As a requlting, the prospective user has no way of evaluating the quality of the ANQR without performing extensive testing.

Some improvement was evidenced in the work of Kahaner,[8] who tested many ANQRs on the same set of 21 test integrals. The same set was used by de Boor[6] for testing his ANQR. At least three of the 21 are not appropriate test integrals, however, because the results are algorithm-dependent in an unrepresentative way.

Two examples will make this clear. First consider

$$\int_0^1 f(x) \cos(\alpha\pi x)\, dx$$

where $f(x)$ is any smooth function. QUAD, which divides according to the $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{6}$ sequence, will fail for α near 36, but not for α near 32. [For α near 36, the regular sampling procedure of QUAD samples only near the peaks of the cosine, so the integrand looks like $f(x)$.] CADRE,[6] which divides according to the $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ sequence, will fail for α near 32, but not for α near 36. A single test integral with a large α may not compare ANQRs fairly. Test integrals 13 and 17 of Kahaner (see Table I) are of this type, each having about 50 full cycles.

Second, consider

$$\sqrt{\alpha} \int_0^1 e^{-\alpha(x-\beta)^2}\, dx$$

for α large and positive. Depending on the choice of β, this can be either easy or hard for a particular ANQR. If the peak comes sufficiently near a sampling point, adaptive ANQRs can zero in on the peak and integrate it accurately, although many sampling points will be necessary. If the peak does not come sufficiently near a sampling point, the integrand looks like zero, and the ANQR fails. For proper comparison of ANQRs, any single α is insufficient. Test integral 21 of Kahaner is of this type.

Table I — Kahaner's 21 Test Integrals

No.	a	b	answer	$f(x)$
Easy				
12	0	1	+0.7775046341	$x/(e^x - 1)$
11	0	1	+0.3798854930	$1/(1 + e^x)$
1	0	1	+1.7182818284	e^x
10	0	1	+0.6931471806	$1/(1 + x)$
4	−1	1	+0.4794282267	$0.92 \cosh(x) - \cos(x)$
8	0	1	+0.8669729873	$1/(1 + x^4)$
5	−1	1	+1.5822329637	$1/(x^4 + x^2 + 0.9)$
20	−1	1	+1.5643964441	$1/(x^2 + 1.005)$
Steeply decaying				
15	0	10	+1.0000000000	$25e^{-25x}$
14	0	10	+0.5000002112	$\sqrt{50}\exp(-50\pi x^2)$
16	0	10	+0.4993638029	$50/[p(1 + 2500x^2)]$
Singular				
6	0	1	+0.4000000000	$x^{3/2}$
3	0	1	+0.6666666667	$x^{1/2}$
2	0	1	+0.7000000000	$0, x < 0.3; 1, x > 0.3$
7	0	1	+2.0000000000	$0, x = 0; x^{-1/2}, x > 0$
19	0	1	−1.0000000000	$0, x = 0; \ln(x), x > 0$
Oscillatory				
18	0	P	+0.8386763234	$\cos[\cos x + 3\sin x + 2\cos 2x + 3\sin 2x + 3\cos 3x]$
9	0	1	+0.4794282267	$2/[2 + \sin(10px)]$
17	0.01	1	+0.1121395696	$50[\sin(50px)/(50px)]^2$
13	0.1	1	+0.0090986453	$\sin(100px)/px$
Isolated peak				
21	0	1	+0.2108027354	$\operatorname{sech}^2[10(x - 0.2)] + \operatorname{sech}^4[100(x - 0.4)] + \operatorname{sech}^6[1000(x - 0.6)]$

Note: $p = 3.14159$, $P = 3.1415927$.

Kahaner did not test noisy functions, and did not ask for impossibly small error tolerances.

3.1 Testing on the Kahaner 21

The Kahaner 21 are listed in Table I. They have been grouped according to type; within groups they are ordered approximately by difficulty. Tables II, III, and IV summarize the results of ten ANQRs on the 21 test integrals, for requested error tolerances 10^{-3}, 10^{-6}, and 10^{-9}. In each table, the first column is the integral number; succeeding columns are the number of sampling points used by each ANQR. An F indicates that the ANQR was unsuccessful, and an asterisk that the ANQR used the fewest sampling points of any successful routine. (For these tables only, "successful" means that the true error of an integration is no more than 20% higher than the requested error, since some ANQRs failed by a small amount.) Columns labeled ROMB through RBUN are based on the number of sampling points reported by Kahaner[8] for seven of his highest-quality ANQRs, and were obtained on a CDC 6600. Column CADRE is from de Boor,[6] and the integrals were performed on a CDC 6500. Columns QSUBA and DQUAD were computed especially for this com-

Table II — Number of sampling points used by each ANQR; attempted absolute accuracy 10^{-3}

Integral	ROMB	SIMPSN	SQUANK	QNC7	QNC10	QABS	RBUN	CADRE	QSUBA	DQUAD
12	17	19	9	25	37	13	5*	9	7	13
11	17	19	9	25	37	13	5*	5*	7	13
1	17	19	9	25	37	13	5*	9	7	13
10	17	19	9	25	37	13	5*	9	7	13
4	17	19	9	25	37	13	5*	17	7	13
8	17	19	9	25	37	13	5*	9	7	15
5	17	19	9*	25	37	13	11	33	15	17
20	17	31	9*	25	37	13	11	17	15	19
15	513	103	53	85	109	85	527	88	63	52*
14	1025	103	49*	97	127	85	51	62	3F	52
16	2049	115	53	121	163	109	87	81	127	48*
6	17	19	9	25	37	13	5*	9	7	15
3	65	55	9F	49	55	77	211	17	15*	17
2	257F	115	29F	121	163	141	271	53*	771	85
7	8193F	235	105F	241	361	133F	211	33	517	26*
19	4097	175	45	217	307	181	211	137	31	28*
18	129	139	53*	85	73	77	39F	107	63	61
9	33*	163	81	97	145	149	79	183	127	101
17	1025	151	57*	165	307	149	109	512	255	185
13	1025	19F	429	49F	865	573	533	1028	255*	381
21	4097	127	17F	97	127	77	65	108	333F	49*

* = best of all successful results
F = failure

parison, on a Honeywell 6070. Double precision was used so that the relative machine precision would be comparable to that of the CDC machines. QSUBA[14] has provision only for relative error; for these tests a relative error was requested which gave the appropriate absolute error request.

It is important to notice that some of the failures are due to an ANQR deciding to stop because of excessive sampling; these failures are far less reprehensible than the others because an error return could have been made, and an incorrect answer rejected. For RBUN through ROMB, this information can only be inferred since Kahaner did not list any error returns. No such failure occurred for CADRE or DQUAD; QSUBA has no built-in maximum.

ROMB[1] is a standard Romberg extrapolation routine, using the standard 1, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ sequence. It is not adaptive. It stops, apparently, after 8193 sampling points of $f(x)$. Thus only one of its failures is serious. ROMB requires at least 17 points before believing any answer.

SIMPSN[5] and SQUANK[11] are adaptive Simpson's rule routines, apparently with cutoffs at 5003 and 5001 points, respectively. These routines are decent at low accuracies, but are not of high-enough order to be competitive at high accuracy. SQUANK also assumes that an improper convergence rate is due to noise in the function, rather than to a singu-

Table III — Number of sampling points used by each ANQR; attempted absolute accuracy 10^{-6}

Integral	ROMB	SIMPSN	SQUANK	QNC7	QNC10	QABS	RBUN	CADRE	QSUBA	DQUAD
12	17	19	9	25	37	13	5*	9	7	13
11	17	19	9	25	37	13	11	9	7*	13
1	17	55	17	25	37	13	21	17	7*	19
10	17	55	21	25	37	13*	31	17	15	19
4	17	55	25	25	37	25	5F	33	15*	19
8	33	67	29	25	37	25	41	17	15*	21
5	65	163	65	49	37	49	59	49	31*	41
20	65	163	49	49	37	49	49	33	31*	33
15	2049	343	213	133	145	133	4117	140	63*	98
14	2049	331	169	133	163	109	91	89	3F	86*
16	8193F	511	273	181	181	145	141	145	255	96*
6	129	91	29*	61	73	65	383	65	31	34
3	4097	199	105	157	217	145	423	33	63	32*
2	8193F	235	29F	241	361	261	271	119*	3351	125
7	8193F	1027F	1153F	241F	361F	89F	587F	129	5925	40*
19	8193F	499	257	241	361	105F	403	233	795	38*
18	257	547	301	181	199	205	195	177	63*	117
9	129*	871	377	289	397	313	267F	409	255	285
17	2049	2275	697F	385F	1009	829	697	1237	255*	547
13	2049	19F	2549	1525	1639	1449	2383	1449	255*	757
21	8193F	691	185F	205F	253F	197F	327*	189F	525F	127F

* = best of all successful results
F = failure

larity, and so quits early on some of the test integrals. SIMPSN requires at least 19 points, and SQUANK 9.

QNC7 and QNC10[8] are adaptive Newton-Cotes routines, with 7- and 10-point rules, respectively. (QNC10 was called QUAD in Ref. 8.) They performed quite well, failing only on some of the most difficult integrals. QNC7 requires at least 25 points, and QNC10 at least 37, somewhat excessive for the easiest integrals.

QABS[13] combines Romberg and Curtis-Clenshaw quadrature, and performed quite well, failing only on some of the most difficult integrals. It requires at least 13 points.

RBUN[4] is an adaptive Romberg extrapolation routine, using the standard sequence. It apparently has a cutoff at 5001 points. RBUN requires at least 5 points, and seems to be somewhat unreliable.

Kahaner recommended any of QNC7, QNC10, or QABS as a library routine. He did not have CADRE, QSUBA, or DQUAD available to test.

CADRE[6] is more recent than the ANQRs just discussed. It uses a version of cautious Romberg extrapolation based on the standard sequence. It also includes provision for recognizing a singularity of the form x^α, estimates α numerically, and extrapolates using the estimated α. CADRE requires at least 5 points. On the test integrals, it seems somewhat more efficient than QNC7, QNC10, and QABS on nonsingular integrals and

Table IV — Number of sampling points used by each ANQR; attempted absolute accuracy 10^{-9}

Integral	ROMB	SIMPSN	SQUANK	QNC7	QNC10	QABS	RBUN	CADRE	QSUBA	DQUAD
12	17	55	33	25	37	13*	39	17	15	19
11	33	151	33	25	37	25	39	17	15*	19
1	17	163	65	25	37	25	73	17	15*	23
10	65	271	97	37	37	49	123	33	15*	25
4	33	331	105	25	37	85	139	33	15*	25
8	65	463	149	73	73	97	129	65	31*	45
5	129	487	289	97	73	181	239	129	31*	73
20	129	487	249	97	73	145	185	129	31*	49
15	4097	1483	1145	241	217	281	5001F	215	127*	138
14	4097	1123	797	241	253	245	259	202	3F	154*
16	8193F	2467	1649	397	343	397	1435	337	255	192*
6	2049	427	161	133	163	137	1423	529	63	46*
3	8193F	883	513	289	361	289	1595	129	255	42*
2	8193F	235	29F	241F	361F	381	271	173	5931	165*
7	8193F	4279F	5001F	589F	685F	89F	2467F	625	11325	70*
19	8193F	2203	1969F	421F	415F	89F	1571	369F	3495	80*
18	513	2923	1589	409	343	589	753	417	127*	217
9	257*	3967	2525	697	757	893	883	785	765	473
17	4097	5003F	5001F	1345F	1999	2025	2741	2329	255*	1109
13	4097	5003F	5001F	3073	2773	3197	5001F	3505	255*	1161
21	8193F	3751	1657	709	685	633*	1079	661	827F	261F

* = best of all successful results
F = failure

much more efficient on singular ones. In addition, the cautious extrapolation means that CADRE's error estimation procedure has some rationale behind it, and CADRE is more robust than the aforementioned routines. However, CADRE is difficult to understand and to maintain, since its style is the antithesis of structured programming. It is one large program, with no subprograms, but with a liberal and unstructured use of GOTOs.

QSUBA[14] uses a series of 8 whole-interval quadrature rules of increasing order, starting with the 1- and 3-point Gauss-Legendre rules. Succeeding rules are constructed to be of as high order as possible, consistent with using all the previous sampling points. The highest-order rule uses 255 points and is of order 383. If convergence is not obtained after 8 rules, the interval is divided in half, and each half considered anew. Unlike all the other ANQRs under consideration, all function values must be discarded, since none of the sampling points on the half intervals coincides with any on the full interval. QSUBA works well on any integral which can be integrated without dividing the interval, and poorly on integrals which require dividing. It is especially good on easy and oscillatory integrals, since a high-order rule is generally used. QSUBA uses at least 3 points, which is somewhat unsafe, but has *no* maximum built in—it goes on forever, if necessary.

DQUAD takes at least 13 points. It fails only on number 21, for high accuracy, missing the narrow peak at $x = 0.6$. The integral which is the same as 21 except for moving the peak to 0.61 is done properly, using 117, 241, and 447 points for error tolerances 10^{-3}, 10^{-6}, and 10^{-9}, respectively. DQUAD is clearly more efficient and robust, based on these test integrals, than any other ANQR tested except QSUBA. DQUAD is more robust than QSUBA, but is less efficient for integrals where QSUBS does not need to divide the interval.

3.2 Parameter studies (1)

Testing an ANQR on a "random" set of test integrals, while instructive and a good start, is insufficient for a library routine. The testing of an ANQR is incomplete without numerous parameter studies:

$$\int_a^b f(x;\alpha)\,dx$$

with fixed ϵ and varying α (Ref. 12), and with fixed α and varying ϵ. The function $f(x,\alpha)$ should be increasingly difficult to integrate as α approaches some limit, and α should be pushed close enough to that limit so that failure occurs. For error tolerance studies, the requested error should range from the approximate value of the integral to less than typical roundoff on the computer being used.

Several of the first type of parameter study will now be discussed. For all of them, the error requested is 10^{-6}. The first was suggested by de Boor[6]:

$$\int_0^1 \frac{2^\alpha}{1 + (2^\alpha x)^2}\,dx$$

For large α, the integrand is highly peaked. The integrand is 2^α at $x = 0$, falls to half that at $x = 2^{-\alpha}$, and is $2^{-\alpha}$ at the endpoints of the interval. There is no danger in missing the narrow spike, since it is exactly at the center of the interval, a normal sampling point for almost all ANQRs. This example demonstrates the power of adaptive ANQR, in that the number of sampling points increases only as $\sqrt{\alpha}$, approximately. The behavior of DQUAD is shown in Fig. 1. The number of sampling points is plotted against α. The meaning of the symbols used for plotting in all the figures is given below.

- · Successful integration; no error flag
- × Unsuccessful integration; error flag
- o Unsuccessful integration; no error flag
- ⊗ Successful integration; error flag

In testing DQUAD, "successful" means that the true error is less than ϵ. DQUAD failed for $\alpha > 38$, but recognized its failure (failure was due to

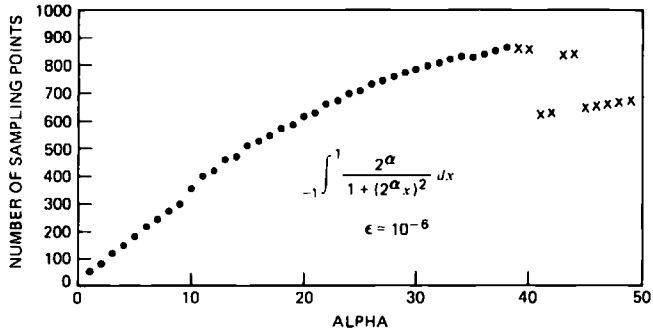

Fig. 1—Performance of DQUAD on function highly peaked at center of interval.

filling of the function value stack). For comparison, CADRE fails starting at $\alpha = 31$, and generally uses about 20 percent fewer sampling points than DQUAD.

This integral should not pose any serious difficulty to a decent ANQR, since the unpleasant behavior occurred exactly at the center of the interval. Usually, though, the user should strive to break up integrals so that any unpleasant behavior happens at one of the endpoints of the interval. It is feasible for ANQRs to recognize such behavior at the endpoints, but difficult if it occurs in the center of the interval. As an example, the previous integral, except with limits 0 and 1, may be considered. DQUAD's performance is shown in Fig. 2. For $\alpha \geq 7$, DQUAD recognizes that the integrand approximates a step function, and makes a change of variable. This change of variable keeps the number of sampling points from growing significantly as α increases.

Figure 3 shows a similar integral, except more steeply decaying away from the endpoint.

$$\int_0^1 2^\alpha e^{-2^\alpha x}\,dx$$

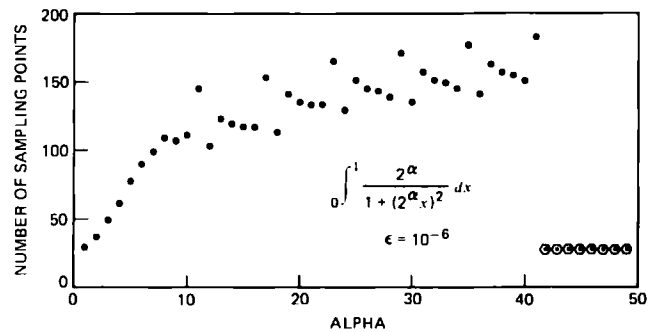

Fig. 2—DQUAD's performance on function highly peaked at end of interval.

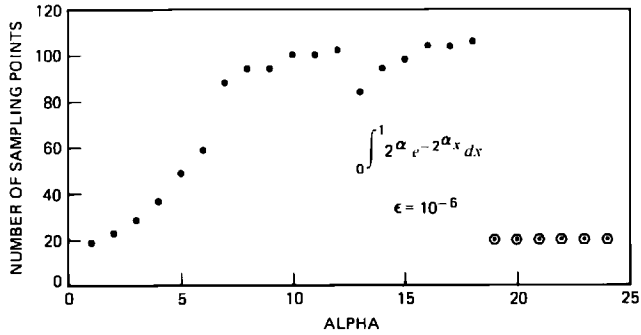

Fig. 3—Performance of DQUAD on steeply decaying function.

Failure eventually occurs, as for the previous integral, when, after the change of variable, the new integrand is a sharply-peaked function with its peak away from an endpoint, and the peak is missed entirely.

Testing of ANQRs on functions with isolated peaks within the range of integration takes more work; α must parameterize the narrowness of the peak, but the position of the peak is also important. A suitable test integral has two parameters. DQUAD was tested on

$$\int_0^1 2^\alpha e^{-4^\alpha(x-\beta)^2}\,dx$$

For each α, 25 integrals were done, with β's of 0.02(0.02.)0.50, for a statistical evaluation. No failures occurred for $\alpha = 1, 2, 3, 4, 5$, or 6; one occurred for $\alpha = 7$, at $\beta = 0.04$. For $\alpha = 8$, 12 out of 25 failed. Figure 4 illustrates the results for $\beta = 0.40$, a typical value.

Another standard test integral, also used by de Boor, is

$$\int_0^1 x^\alpha\,dx$$

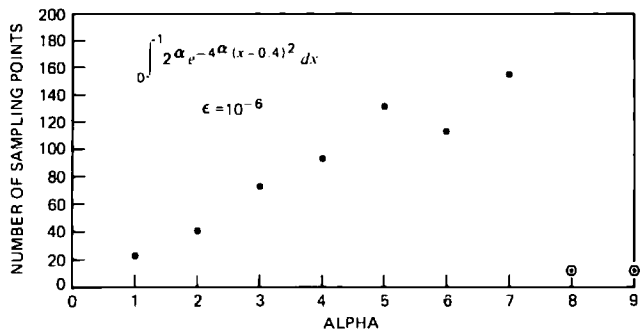

Fig. 4—Performance of DQUAD on highly peaked function.

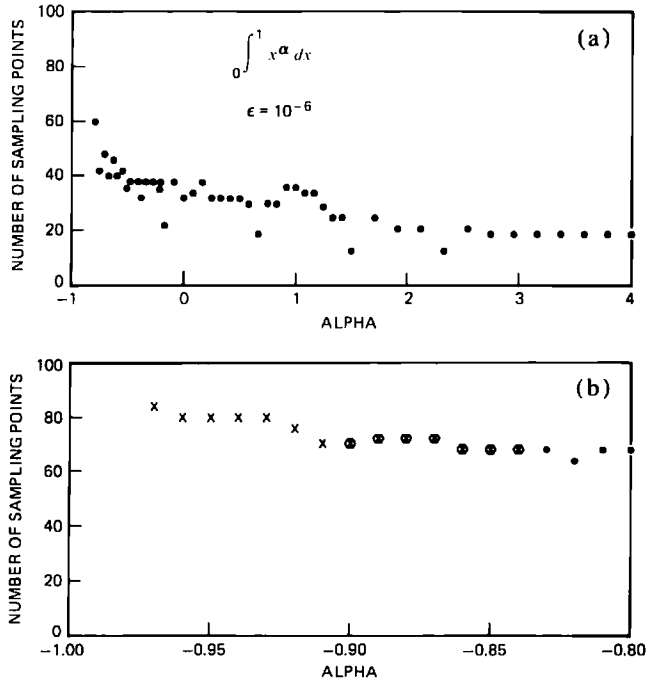

Fig. 5—Performance of DQUAD on function with endpoint singularity.

with the integrand set equal to zero at $x = 0$. Thus for $\alpha = 0$ the integrand is a step function, and for $\alpha < 0$ the integrand has an infinite discontinuity. For $\alpha < 0$, the integrand is "unreasonable" by almost anyone's definition, but users sometimes give such integrals to ANQRs. Figure 5 shows the performance of DQUAD. A change of variable was made automatically by DQUAD for $-0.97 \leqslant \alpha < 1.75$, approximately. DQUAD is designed to reject the change of variable if the estimated α is less than -0.99, and the change is not necessary to achieve the desired accuracy for $\alpha \geqslant 1.75$. For $\alpha < -0.97$, DQUAD fails with an error flag, using about 900 function samples. Machine precision limits the efficiency of the change of variable for α less than about $-3/4$. For comparison, CADRE fails, with an error flag, for α less than $-7/8$, and gives an erroneous error flag for α near, but not at, $\alpha = 1$. CADRE is substantially less efficient than DQUAD for this test integral. Other ANQRs, without special procedures for endpoint singularities, are much worse.

A final type of test integral is one with an oscillatory integrand. Since almost all ANQRs sample the integrand at regularly spaced points, there is a danger of undersampling. As an example, consider

$$\int_0^1 [1 + \cos(\alpha \pi x)] \, dx$$

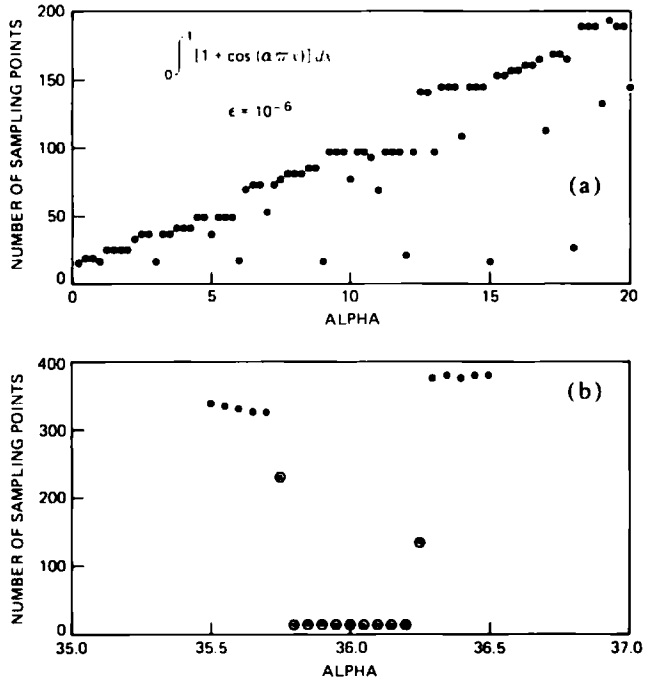

Fig. 6—Performance of DQUAD with oscillatory integrand.

as suggested by de Boor. Figure 6 illustrates the performance of DQUAD on this test integral. The first failure is for α near 36, where the regular sampling causes the cosine to look like unity. The lower part of the figure shows the region near 36. The top part shows the number of sampling points used for α from $1/4$ to 20, with spacing of $1/4$. The trend line is approximately $N = 10\alpha$; CADRE's trend line is approximately $N = 20\alpha$. Besides the general rising trend, there many dots significantly below the trend line. These occur because of resonance between the regular sampling of DQUAD and the regular oscillation of the integrand. If α is integral, or very nearly so, coarse sampling may indicate that the integrand is simpler than it really is. For example, if α is an odd integer, the cosine is odd about $x = 0.5$, and the center third of the integral is integrated (correctly) based on insufficient sampling, since the trapezodial rule correctly integrates odd functions.

This kind of resonance phenomenon is a difficult problem for an ANQR to surmount. The best way is probably to back off slightly from the notion of a fully automatic ANQR. Usually a prospective user of an ANQR knows if an integrand is oscillatory, and further, knows the approximate period of the oscillation. To avoid problems with possible undersampling, the user could divide the integral into several integrals, each with only

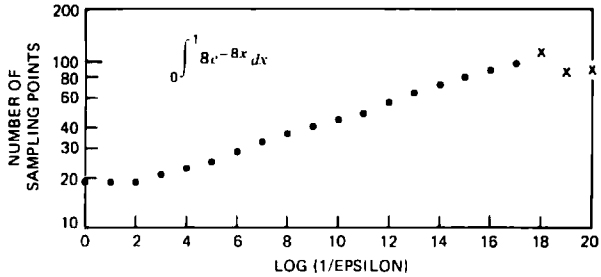

Fig. 7—Parameter test with accuracy varied.

a few periods of the oscillation. An easier way is for the user to require the ANQR to sample the integrand sufficiently finely to see all the oscillations; this assumes the user knows the approximate period. DQUAD has provision for this, using the parameter HSAMPL, discussed in Section IV. If HSAMPL is taken to be $1/\alpha$ in the above example, all the dots fall on the main trend line, and no failures occur.

3.3 Parameter studies (2)

The second type of parameter testing, using a single test integral and varying the requested error, will be considered now. Some ANQRs will fail when the requested error is large compared to the value of the integral, because of insufficiently cautious error estimation. All ANQRs fail if the requested error tolerance is too small, because of roundoff. A properly designed library ANQR will have some mechanism for dealing with too-small error tolerances. Finally, the graph of the number of sampling points versus the requested accuracy is of interest.

Figure 7, for

$$\int_0^1 8e^{-8x}\,dx$$

is typical of DQUAD's performance on easy integrals. The graph of N against $\log 1/\epsilon$ is horizontal at large requested error, with no failures, since DQUAD does not believe any answer until it can be confident of the error estimate. The central portion of the graph is roughly linear. The number of sampling points needed is approximately proportional to $\epsilon^{-1/20}$. DQUAD is effectively functioning as a 20th-order method. The smallest-error part of the graph is also horizontal; accuracy is limited by roundoff.

Figure 8, for

$$\int_0^1 x^{1/2}\,dx$$

is similar except for the smallest requested errors. A change of variable

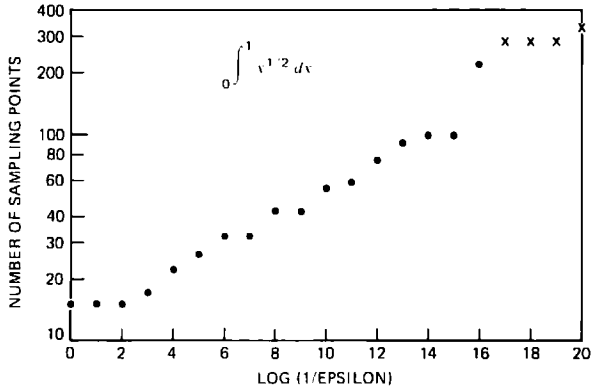

Fig. 8—Parameter test with accuracy varied.

is made only for $\epsilon = 10^{-4}$ through 10^{-16}. For $\epsilon = 10^{-2}$ through 10^{-15}, DQUAD is effectively a 15th-order method. The effective order is less than for the previous example because only three columns of the T table can be asymptotic. Roundoff begins to pollute the answer at $\epsilon = 10^{-16}$.

Figure 9, for

$$\int_0^1 [1 + \cos(\alpha\pi x)]\, dx$$

for two values of α, 1.95 and 17.95, is the last example. For $\alpha = 1.95$, DQUAD is effectively a 20th-order method for $\epsilon = 10^{-3}$ through 10^{-17}. For $\alpha = 17.95$, DQUAD does not begin to be a 20th-order method until $\epsilon = 10^{-6}$.

3.4 Parameter studies (3)

The final type of parameter testing uses a single test integral and error tolerance, but varies the amount of noise in the function. For testing,

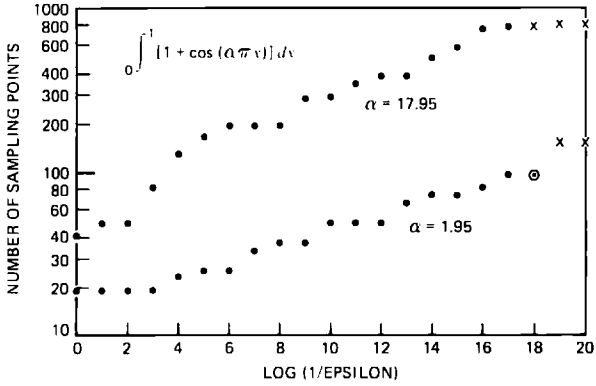

Fig. 9—Parameter test with accuracy varied.

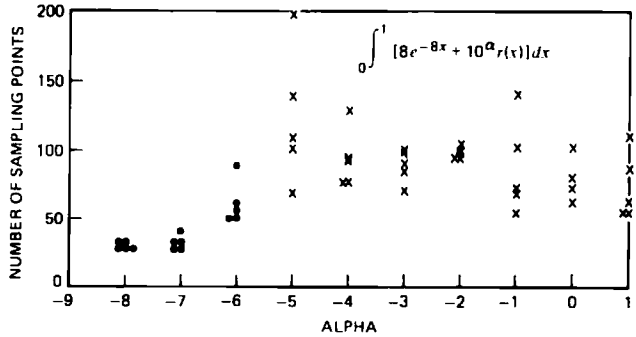

Fig. 10—Parameter test with amount of noise varied.

integrals of the types

$$\int_0^1 [f(x) + 10^{\alpha} r(x)]\,dx$$

$$\int_0^1 f(x)[1 + 10^{\alpha} r(x)]\,dx$$

were done, with requested accuracy $\epsilon = 10^{-6}$. The function $r(x)$ is the output of a pseudorandom number generator with values uniformly distributed on $(-1,1)$. Values used for α were 1, 0, -1, . . . , -8. For f, the same four functions were used as in the previous section. For each function and each α, five different starting points of the random number generator were used. Sample results are shown in Fig. 10 and 11. The number of function values used is plotted vs. α, for each of the five tries. (Where two or more of the tries coincide, they are plotted side by side.) Points plotted with · are those for which DQUAD returned with an error estimate less than 10^{-6}; points plotted with × indicate an error estimate greater than 10^{-6}.

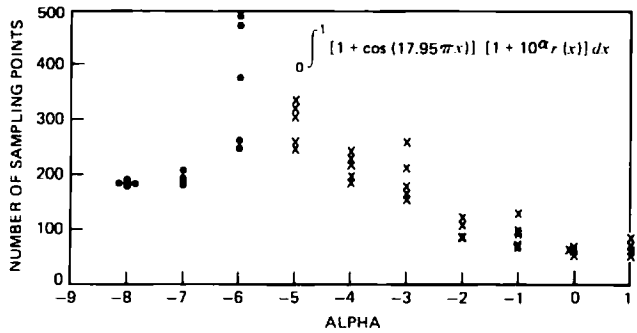

Fig. 11—Parameter test with amount of noise varied.

A summary of the results follows.

For every test with $10^{\alpha} > 10^{-6}$, DQUAD recognized the presence of noise, estimated its magnitude, adjusted accuracy tolerances accordingly, and gave an answer and error estimate. The error estimates given were at most twice 10^{α}, and were generally less than 1.2 times 10^{α}. In only 6 out of 280 integrals was the calculated answer farther from the noise-free value of the integral than the estimated error returned.

For every test with $10^{\alpha} = 10^{-6}$, DQUAD returned an answer with an error estimate less than 10^{-6}. In 3 out of 40 integrals, the presence of noise was recognized. In 2 out of 40, the calculated answer was between 1 and 2 times 10^{-6} away from the noise-free value; in the remaining 38, the answer was less than 10^{-6} away.

For every test with $10^{\alpha} < 10^{-6}$, DQUAD returned an answer with an error estimate of less than 10^{-6}, and the calculated answer was less than 10^{-6} away from the noise-free value. No noise was recognized.

IV. IMPLEMENTATION OF QUAD

QUAD is a Fortran subroutine which attempts to evaluate

$$\int_a^b f(x)\,dx$$

to within absolute accuracy ϵ; the user supplies a, b, ϵ, and a Fortran function subprogram to evaluate $f(x)$. The discussion also applies to the double precision version, DQUAD, except as noted.

QUAD is written in PFORT,[15] a portable subset of ANS Fortran. Temporary storage space is managed by a portable Fortran stack mechanism.[7]

The calling sequence is

```
CALL QUAD (F,A,B,EPS,ANS,ERREST)
```

F is the name of the user's subprogram, A and B are the limits, and EPS the requested accuracy. The estimated value of the integral and an estimate of the accuracy of the estimate are given in ANS and ERREST. If ERREST is larger than EPS, QUAD invokes the PORT library's centralized error handling facility,[7] turning on the error state before returning control to the calling program. If the user has not taken prior action to recover from errors, an error message is printed and the run is terminated. The user who has taken prior action may test for the error state, and continue if desired.

QUAD sets seven parameters to their default values and then calls R1QUAD. The default values should be adequate for almost all users, but the user wishing other values may call R1QUAD directly. R1QUAD also gives more information to the user about problems encountered during the integration. R1QUAD's calling sequence is

CALL R1QUAD(F,A,B,EPS,HSAMPL,NFCALL,LYSTAK,KMAXEX,
KDIVID,JPRINT,NUMINT,ANS,ERREST,KWARN)

The default parameter values used by QUAD are

HSAMPL	0.125
NFCALL	2000
LYSTAK	250
KMAXEX	6
KDIVID	4
JPRINT	0
NUMINT	3

The same parameters are used for DQUAD, except that KMAXEX = 8.

HSAMPL measures how finely $f(x)$ must be sampled. No error estimate is believed unless the trapezoidal rule step size is HSAMPL $|b - a|$ or smaller. Reducing HSAMPL improves the robustness of R1QUAD, but decreases its efficiency. Changing HSAMPL is useful for integrating oscillatory functions, where there is some danger of aliasing. For $\int_0^1 \cos(\alpha\pi x)\, dx$, HSAMPL $= 1/\alpha$ is safe. The minimum number of sampling points of $f(x)$ is roughly 2/HSAMPL.

NFCALL is the upper limit on the number of sampling points of $f(x)$.

LYSTAK is the length of the stack for storing values of the function at the sampling points. Space for the stack is allocated within R1QUAD using a portable storage-allocation package.[7]

KMAXEX is the maximum number of extrapolations allowed. Legal values are 4, 6, 8, 10, and 12.

KDIVID is the minimum number of extrapolations required before dividing an interval. Legal values are 4, 6, 8, . . . , KMAXEX.

JPRINT determines how much printing will be done. With JPRINT < 1, there is none. With JPRINT = 1, the endpoints, attempted error tolerance, answer, and error estimate are printed for each interval attempted. With JPRINT > 1, the T tables are also printed.

NUMINT is the number of equal intervals into which $[a,b]$ is divided to start. If NUMINT > 1, signularities can be recognized at $x = a$ and at $x = b$; if NUMINT = 1, a singularity can be recognized only at $x = a$. Increasing NUMINT generally increases robustness while decreasing efficiency.

KWARN is an integer warning flag, output from R1QUAD. It is zero if all went well. If KWARN is positive, it may have up to 6 digits, each with an independent meaning. Although ERREST may be greater than the requested EPS, it is still reliable. Each digit is zero unless a problem occurred; starting with the right-most digit, the problems are:

(*i*) The error estimate is limited by noise or roundoff, but is above the requested error.

(*ii*) The interval size is as small as is allowed.

(*iii*) As many intervals are stacked as is allowed.

(*iv*) As many function values are stacked as is allowed.

(*v*) As many function sampling points have been used as is allowed.

(*vi*) The error estimate is larger than the requested error.

4.1 Machine-dependent constants

All machine dependency is isolated into four machine-dependent constants, which are set by R1QUAD. No reprogramming is necessary to run QUAD or R1QUAD on another computer. The constants are set using the portable machine constants program R1MACH of Ref. 7.

DLARGE is used as error estimate before any error estimates are considered believable. Its value is set to slightly less than the largest floating-point number, 0.1 R1MACH(2).

DSMALL is used as the default magnitude of $r_2(x)$. Its value is slightly larger than the smallest positive floating-point number, 10 R1MACH(1).

DROUND is used as the default magnitude of $r_1(x)$. Its value is set to 50 times the largest relative rounding error, or 50 R1MACH(4).

HSMALL is the smallest fraction of $|b - a|$ used for a trapezoidal rule step size. Its value is the same as DROUND.

REFERENCES

1. F. Bauer, H. Rutishauser, and E. Stiefel, "New Aspects in Numerical Quadrature," Proc. of Symposia in Applied Mathematics, *15*, pp. 199–218, American Mathematical Society, Providence, 1963.
2. R. Bulirsch and J. Stoer, "Fehlerabschaetzungen and Extrapolation mit Rationalen Funktionen bei Verfahren vom Richardson-Typus," Num. Math., *6*, 1964, pp. 413–427.
3. R. Bulirsch and J. Stoer, "Asymptotic Upper and Lower Bounds for Results of Extrapolation Methods," Num. Math., *8*, 1966, pp. 93–104.
4. W. Bunton, M. Diethelm, and K. Haigler, "Romberg Quadrature Schemes for Single and Multiple Integrals," Jet Propulsion Laboratory Report TM-324-221, 1969.
5. P. J. Davis and P. Rabinowitz, *Numerical Integration*, Waltham, Mass.: Blaisdell, 1967.
6. C. de Boor, "CADRE: An Algorithm for Numerical Quadrature," in *Mathematical Software*, J. R. Rice (ed.), New York: Academic Press, 1971, pp. 417–449.
7. P. A. Fox, A. D. Hall, and N. L. Schryer, "The PORT Mathematical Subroutine Library," Bell Laboratories Computer Science Technical Report No. 47, September, 1976. Accepted for publication in ACM Transactions on Mathematical Software. Inquiries about the PORT library may be addressed to Bell Laboratories Computing Information Service, 600 Mountain Avenue, Murray Hill, New Jersey 07974.
8. D. K. Kahaner, "Comparison of Numerical Quadrature Formulas," in *Mathematical Software*, J. R. Rice (ed.), New York: Academic Press, 1971, pp. 229–259.
9. R. E. Lynch, "Generalized Trapezoidal Formulas and Errors in Romberg Quadrature," Blanch anniversary volume, Aerospace Research Laboratories, Office of Aerospace Research, United States Air Force, pp. 215–229, 1967.

10. J. N. Lyness and B. W. Ninham, "Numerical Quadrature and Asymptotic Expansions," Math. Comp., *21* (1967), pp. 162–178.
11. J. N. Lyness, "Algorithm 379, SQUANK," Comm. ACM, *13*, 1970, pp. 260–263.
12. J. N. Lyness, attribution by C. de Boor (Ref. 6).
13. H. O'Hara and F. Smith, "The Evaluation of Definite Integrals by Interval Subdivision," Nat. Bur. of Standards, Report N69-11541, 1969.
14. T. N. L. Patterson, "Algorithm 468, Algorithm for Automatic Numerical Integration over a Finite Interval," Comm. ACM, *16*, 1973, pp. 694–699.
15. B. G. Ryder, "The FORTRAN Verifier: User's Guide," Bell Telephone Laboratories Computing Science Technical Report No. 12, March 1973.