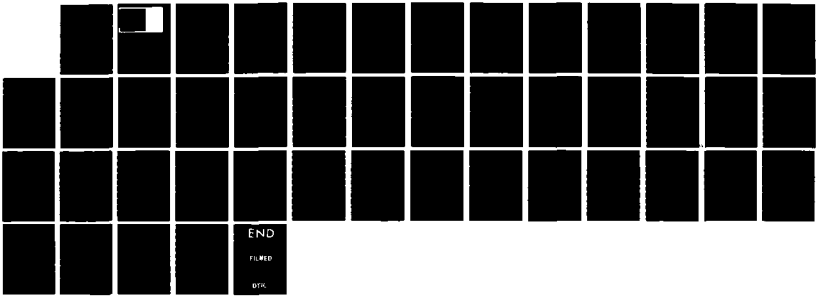
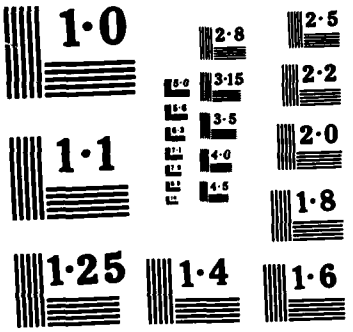
AD-A158 164 ADAPTIVE SELF-VALIDATING NUMERICAL QUADRATURE(U) 1/1
WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER
G F CORLISS ET AL. MAY 85 MRC-TSR-2815 DAAG29-80-C-0041
UNCLASSIFIED F/G 12/1 NL





(2)

AD-A158 164

MRC Technical Summary Report #2815

ADAPTIVE, SELF-VALIDATING NUMERICAL QUADRATURE

George F. Corliss and L. B. Rall

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

May 1985

(Received May 6, 1985)

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

UNIVERSITY OF WISCONSIN-MADISON
MATHEMATICS RESEARCH CENTER

ADAPTIVE, SELF-VALIDATING NUMERICAL QUADRATURE

George F. Corliss and L. B. Rall

Technical Summary Report #2815
May 1985

ABSTRACT

Integrals of a function of a single variable can be expressed as the sum of a numerical quadrature rule and a remainder term. The quadrature rule is a linear combination of function values and weights, or the integral of a Taylor polynomial, while the remainder term depends on some derivative of the integrand evaluated at an unknown point in the interval of integration. Numerical quadrature is made self-validating by using interval computation to capture both the roundoff and truncation errors made when using a given rule. Necessary derivatives can be generated automatically by using well-known recurrence relations for Taylor coefficients. In order for quadrature methods of this type to be accurate (in the sense that small intervals containing the exact result are produced) and efficient (to obtain results of given accuracy in a reasonably short time), an accurate scalar product and an adaptive strategy are required. The necessary scalar product and support for interval arithmetic are provided in Pascal-SC (for microcomputers) and ACRITH (for IBM 370 computers). The adaptive strategy chooses the subintervals of integration and the order of the quadrature formula used in each subinterval on the basis of guaranteed, rather than estimated, information about the error of the numerical integration in each subinterval. The program described in this report implements standard Newton-Cotes, Gaussian, and Taylor series methods for numerical integration. Ways to handle singularities are discussed, and comparisons are given with a standard numerical integration method.

AMS (MOS) Subject Classifications: 65D30, 65G10

Key words: Numerical quadrature, Guaranteed error bounds, Automatic Differentiation, Interval computation

Work Unit Number 3 - Numerical Analysis and Scientific Computing

Sponsored in part by the U. S. Army Under Contract No. DAAG29-80-C-0041.

SIGNIFICANCE AND EXPLANATION

Routines for numerical integration are among the most heavily used programs in computer libraries supporting scientific, engineering, and statistical computation. The ones in common use at the present time, such as CADRE and QUADPACK, produce approximations to integrals with only estimates of the error in the value returned. The program described in this report performs self-validating numerical quadrature, returning an interval in which the desired integral is guaranteed to lie. To do this, automatic differentiation and interval computation are used to capture both the roundoff and truncation error inherent in standard Newton-Cotes, Gaussian, and Taylor polynomial methods for numerical integration. The actual integration can be expressed as the interval scalar product of a vector of function values with a vector of weights, so in order to obtain accuracy (small intervals) the computation has to be supported by an accurate scalar product as well as interval arithmetic. Fortunately, these capabilities are provided in Pascal-SC for microcomputers, and ACRITH for IBM 370 mainframe computers. The program described in this report is written in FORTRAN, and uses ACRITH. This avoids the kind of special implementation of interval computation which was necessary on outdated systems. Since the IEEE standard for floating-point arithmetic requires the support of interval arithmetic, it can be assumed that self-validating computation in general will be much easier to perform as conforming systems become available. Comparisons of the program described in this paper with the standard integration routine QUADPACK indicate comparable execution times; however, the results of QUADPACK lack any guarantee of accuracy.

ADAPTIVE, SELF-VALIDATING NUMERICAL QUADRATURE

George F. Corliss and L. B. Rall

1. Requirements for Automatic Integration Algorithms

In [2], de Boor formulates fundamental requirements for an automatic algorithm for numerical approximation of the integral

$$If = I_{[a,b]}f = \int_a^b f(x)dx \tag{1.1}$$

of a function of a single real variable. Such an algorithm requires (i) the limits of integration a, b, (ii) access to a procedure for the evaluation of $f(x)$ for x in the interval of integration, (iii) tolerances α, ρ on the desired absolute and relative error, respectively, and (iv) a limit M on the number of function evaluations allowed.

As output, the algorithm should produce an estimate I^* for the value of If which satisfies

$$|If - I^*| \leq \max\{\alpha, \rho|If|\}. \tag{1.2}$$

Furthermore, the algorithm should be *efficient*, computing as few function values as possible. It should also be *reliable*, which will be taken here to mean that either the desired accuracy (1.2) is guaranteed, or a message to the contrary is returned to the user, possibly with additional information about the cause of failure. As pointed out by de Boor [2], algorithms which use only values of $f(x)$ at a finite number of points cannot meet the above requirements in general; nevertheless, accurate and efficient automatic integration algorithms can be formulated for wide classes of integrands [3], [23].

This paper presents automatic quadrature algorithms which attain the goals of reliability and efficiency by use of automatic differentiation and interval computation. They

Sponsored in part by the U. S. Army under Contract No. DAAG29-80-C-0041.

make use of information about the integrand on entire subintervals of integration, rather than at a discrete set of points. The results combine the self-validating algorithms of Gray and Rall [8], [9], [10], and the notion of adaptive quadrature [2], [3], [23]. Adaptation is carried out on the basis of guaranteed, rather than estimated, bounds for the error of the approximate integration over each subinterval. Furthermore, the given algorithm has the ability to detect and handle certain types of singularities in the integrand, and even to verify nonexistence of the integral in some cases. In the terminology of Rice [23], the method described here has the following features:

Interval Processor Component:

Variable order rule with remainder using interval arithmetic to give guaranteed bounds.

Bound Estimator Component:

Direct analysis.

Special Behavior Components:

Polynomials.

Roundoff level.

Singularities in derivatives.

Jump discontinuities.

Removable singularities.

x^{α}-type singularities.

All are strictly validated.

Interval Collection Management Component:

Ordered list.

None discarded.

The method of this paper does not belong to the large family of 10^6 or so algorithms considered by Rice because of the use of interval computation and automatic differentiation,

which were not considered in [23]. Details of the actual implementation of the algorithms presented here in an environment which supports interval computation will be given in §7. The next few sections describe the underlying methodology.

2. Self-validating Evaluation of Quadrature Formulas

Self-validation of numerical computations is one of the basic motivations of interval analysis [1], [15], [16]. The goal is to obtain an interval which contains the desired result, be it real or set-valued. In the case of the integration problem (1.1), a self-validating interval method produces an interval $J = [c, d]$ which is guaranteed to contain the value If of the integral. The width of this interval inclusion will depend on uncertainties in the values of the integrand and the limits of integration, the roundoff error in the actual computation, and the truncation error appropriate to the method used. All of these quantities can be estimated in a tedious way by the techniques of classical error analysis, an effort which is unnecessary in the computational environment described below. However, once an interval $[c, d]$ containing I is found by whatever method, one has the following approximations to I and corresponding error bounds [21]:

$$I^* = \frac{1}{2}(c + d), \quad |If - I^*| \leq \frac{1}{2}(d - c), \tag{2.1}$$

for absolute error, or

$$I^* = \frac{2cd}{c + d}, \qquad \left|\frac{If - I^*}{If}\right| \leq \left|\frac{d - c}{c + d}\right|, \tag{2.2}$$

for relative error, with $cd > 0$ in this case. It follows that (1.2) will be satisfied if an interval $J = [c, d]$ can be obtained with width $w(J) = d - c$ small enough so that $w(J) \leq 2\alpha$ and $w(J) \leq \rho|c + d|$.

First, the problem of finding an interval inclusion J of If will be considered. The basic method for interval integration by use of standard formulas for numerical quadrature or Taylor series was first described by Moore [14]. To illustrate Moore's idea, consider a

standard interpolatory integration *formula* of the form

$$\int_a^b f(x)dx = \sum_{i=1}^{n} w_i f(x_i) + c_n h \cdot \frac{f^{(p)}(\xi)h^p}{p!}, \tag{2.3}$$

where $h = (b-a)/n$, and $a < \xi < b$. A formula of type (2.3) will be called a quadrature formula of *order p on n points*. The ordinary Gauss and Newton-Cotes integration formulas follow this pattern [7].

It should be noted that integration formulas such as (2.3) give the *exact* value of the integral If of functions which are differentiable p times. The only difficulty is that the value of ξ is unknown. For practical computation, it is thus customary to express (2.3) as the sum of a *rule*

$$r_n f = \sum_{i=1}^{n} w_i f(x_i) \tag{2.4}$$

of numerical integration, and a (truncation) *error term*

$$e_n f = c_n h \cdot f_p(\xi, h), \tag{2.5}$$

where

$$f_p(\xi, h) = \frac{f^{(p)}(\xi)h^p}{p!} \tag{2.6}$$

denotes the Taylor coefficient of order p in the expansion of $f(\xi + h)$. It is usual to compute $I^* = r_n f$ to approximate the value of the integral, and to estimate $e_n f$ somehow. Of course, if f is a polynomial of degree $p-1$ or less, then $e_n f \equiv 0$, and $If = r_n f$.

A self-validating computation of the rule $r_n f$ of numerical integration is straightforward in an environment which provides interval arithmetic and monotone interval inclusions of the library functions used in the evaluation of $f(x)$ for a given x. Let $\mathbf{S}$ denote the screen of floating-point numbers available, and $\mathbf{IS}$ the corresponding set of closed intervals $[u, v]$, $u, v \in \mathbf{S}$. If f is evaluated on an interval $X \in \mathbf{IS}$ using interval arithmetic and

library functions, then the result is the *natural interval inclusion* $F(X)$ of f on X such that

$$f(X) = \{f(x) \mid x \in X\} \subseteq F(X), \tag{2.7}$$

[15], [16]. If W_i, X_i respectively denote the smallest intervals in **IS** which contain the real numbers w_i, x_i, that is, $W_i = [\nabla w_i, \Delta w_i]$, $[\nabla x_i, \Delta x_i]$, where ∇, Δ denote the monotone downward and upward roundings from the real numbers **R** to **S** [11], then the inclusion

$$r_n f \in R_n f = \sum_{i=1}^{n} W_i F(X_i) \tag{2.8}$$

is guaranteed, and the computation of R_n can be done automatically.

An automatic, self-validating computation of the error term (2.5) requires an additional ingredient. This consists of subroutines for the generation of the Taylor coefficients $f_p(x,h)$ of f. These use well-known recurrence relations for the arithmetic operations and library functions used to evaluate $f(x)$ for given x [6], [15], [16], [19]. A suitable computational environment provides these routines. Corliss and Chang [5] have shown that the calculation of $f_0(x,h)$, $f_1(x,h)$,...,$f_p(x,h)$ requires about

$$t = ap^2 + bp + c \tag{2.9}$$

units of time, where a depends on the number of multiplications, divisons, and calls to library functions in the computation of f, and b depends on the number of additions and subtractions required. In any case, interval evaluation of exactly the same recurrence relations yields the corresponding interval inclusions $F_0(X,H)$,...,$F_p(X,H)$ such that

$$f_k(x,h) \in F_k(X,H), \quad k = 0,1,\ldots,p, \tag{2.10}$$

for all $x \in X$ and $h \in H$ [15], [16]. Thus, the desired interval inclusion

$$e_n f \in E_n f = C_n H \cdot F_p(X,H) \tag{2.11}$$

can be computed automatically, given intervals $C_n, H, X \in \mathbf{IS}$ such that $c_n \in C_n$, $h \in H$, and $[a,b] \subseteq X$. (It is assumed that $a \leq b$; the contrary case can be handled easily.) It follows from the above that

$$If \in R_n f + E_n f = [c,d], \tag{2.12}$$

a formula for automatic, self-validating numerical quadrature. Once again, if f is a polynomial of degree $p-1$ or less, then $F_p(X,H) \equiv [0,0]$, so that $If \in R_n f$, and the width of $[c,d]$ depends only on the roundoff error in the calculation of $r_n f$.

This approach was the basis of an actual computer program [9], which met the accuracy criteria (1.2), if possible, by choosing H sufficiently small [18]. However, the problem of efficiency was not addressed.

Instead of splitting the numerical integration formula (2.3) into a rule of numerical integration (2.4) and an error term (2.5), it will be helpful later to consider it to represent the scalar product of the augmented *function-value vector*

$$\mathbf{f} = \big(f(x_1), \ldots, f(x_n), f_p(\xi,h)\big), \tag{2.13}$$

and the augmented *weight vector*

$$\mathbf{w} = (w_1, \ldots, w_n, c_n h), \tag{2.14}$$

which is independent of the integrand f and depends only on the specific formula (2.3) used. Thus,

$$I = \mathbf{w} \cdot \mathbf{f} \in \mathbf{W} \cdot \mathbf{F} = J, \tag{2.15}$$

where

$$\mathbf{F} = \big(F(X_1), \ldots, F(X_n), F_p(X,H)\big) \tag{2.16}$$

and

$$\mathbf{W} = (W_1, \ldots, W_n, C_n H) \tag{2.17}$$

are the corresponding interval inclusions of f, w. This allows the computation to use recently developed methods for highly accurate calculation of real and interval scalar products [12]. This results in a considerable decrease in width due to roundoff error in the computed value of J.

The integration formula (2.3) can be interpreted as a *single-panel* rule, or as a *multi-panel* rule, meaning that a simpler formula on m points is applied k times to the corresponding number of subintervals of $X = [a, b]$, with $n \leq km$. Denoting the subintervals of X by X_i, $i = 1, 2, \ldots, k$, this means that an integration formula

$$\int_{X_i} f(x)dx = \sum_{j=1}^{m} w_{ij} f(x_{ij}) + c_{im} h_i \cdot f_p(\xi_i, h_i), \tag{2.18}$$

holds in each subinterval, where $h_i = w(X_i)$. It has been shown [18] that

$$\sum_{i=1}^{k} c_{im} w(X_i) F^{(p)}(X_i) \cdot \frac{w(X_i)^p}{p!} \subseteq c_n w(X) F^{(p)}(X) \cdot \frac{w(X)^p}{p!}. \tag{2.19}$$

In addition to the decrease in width of $F_p(X, w(X))$ by a factor of $w(X)^p$ as $w(X)$ becomes small, the width of $F^{(p)}(X)$ will overestimate the width of $f^{(p)}(X)$ by less as $w(X) \to 0$ for $f^{(p)}(x)$ continuous [15]. Thus, the gain in calculating the error terms over smaller subintervals can be substantial. Roundoff error in adding a number of interval inclusions of one-panel rules (2.18) can again be reduced considerably by expressing the result as the scalar product of the extended augmented function-value vector

$$\mathbf{F} = \left(F(X_{11}, \ldots, F(X_{1m}), F_p(X_1, H_1), \ldots, F(X_{k1}), \ldots, F(X_{km}), F_p(X_k, H_k)\right) \tag{2.20}$$

with the extended augmented weight vector

$$\mathbf{W} = \left(w_{11}, \ldots, w_{1m}, C_{1m} H_1 \ldots, w_{k1}, \ldots, w_{km}, C_{km} H_k\right). \tag{2.21}$$

Taylor Series Methods

he seminal paper by Moore [14] also provides the basis for self-validating numerical inte-
ation by the use of Taylor series, although the techniques presented by him in this case
e directed toward the solution of the initial-value problem for ordinary differential equa-
ons. For numerical integration, Taylor series are more appropriate than fixed quadrature
rmulas for interval-valued endpoints of intervals of integration, as will be discussed in §6.
rthermore, Taylor series support the rigorous approach to automatic recognition and
eatment of singularities described in §5.

Of course, one could consider (1.1) to be the solution $If = y(b)$ of the initial-value
oblem

.1) $$y'(x) = f(x), \qquad y(a) = 0,$$

d apply Moore's methods directly. However, since $f(x)$ in (3.1) is independent of y,
like the usual case in differential equations, it is simpler to use the capability to generate
segment of the Taylor series and the interval remainder term automatically to perform
elf-validating calculation of the desired integral.

In particular, instead of expanding the solution $y(x)$ of (3.1) at $x = a$ as in the case
a differential equation, it is advantageous to expand $f(x)$ at the midpoint $c = (a+b)/2$
the interval $X = [a, b]$ of integration. It will be assumed that the integrand f has $p \geq 0$
rivatives in the interval of integration. For $h = (b-a)/2$, one has

$$f(x) = f(c) + f'(c)(x-c) + f''(c)\frac{(x-c)^2}{2!} + \cdots + f^{(n-1)}(c)\frac{(x-c)^{n-1}}{(n-1)!}$$

2)

$$+ f^{(n)}(\xi)\frac{(x-c)^n}{n!},$$

$n \leq p$, where $\xi \in X$ is between c and x. Let $F^{(n)}$ be an interval inclusion of $f^{(n)}$ on
Then, $f^{(n)}(\xi) \in F^{(n)}(X)$, which is an interval-valued constant. From (3.2),

3) $$f(x) \in f(c) + f'(c)(x-c) + \cdots + f^{(n-1)}(c)\frac{(x-c)^{(n-1)}}{(n-1)!} + F^{(n)}(X)\frac{(x-c)^n}{n!}.$$

Let

$$g(x) = f(c)(x-c) + f'(c)\frac{(x-c)^2}{2!} + \cdots + f^{(n-1)}(c)\frac{(x-c)^n}{n!} \tag{3.4}$$

be an indefinite integral of the Taylor polynomial of degree $n-1$ of $f(x)$. Then,

$$\int_a^b f(x)dx \in g(x)\Big|_a^b + F^{(n)}(X)\frac{(x-c)^{n+1}}{(n+1)!}\Big|_a^b \subseteq J_n, \tag{3.5}$$

where

$$J_n = 2\sum_{\substack{i=0 \\ i \text{ even}}}^{n-1} F^{(i)}(C)\frac{H^{i+1}}{(i+1)!} + \tag{3.6}$$

$$+ \begin{cases} \left[F^{(n)}(X)\dfrac{H^{n+1}}{(n+1)!} - F^{(n)}(X)\dfrac{H^{n+1}}{(n+1)!}\right], & \text{for } n \text{ odd}, \\ 2F^{(n)}(X)\dfrac{H^{n+1}}{(n+1)!}, & \text{for } n \text{ even}. \end{cases}$$

Note that subtraction does not "cancel" equal intervals in general. One has $[u,v]-[u,v] = [u-v, v-u] \neq [0,0]$ unless $u=v$, in which case the interval consists of a single point [1], [15], [16]. If the series were expanded at $x=a$ instead of $x=c$, then the width of the interval remainder terms would be increased by a factor of 2^{n+1}.

Formulas (3.5)-(3.6) resemble (2.12) for ordinary quadrature rules, with the evaluation of the integrand at n points replaced by its value and the values of its first $n-1$ derivatives at a single point. If f is a polynomial of degree $n-1$ or less, then $F^{(n)}(X) \equiv [0,0]$, and only roundoff error effects the width of J_n.

J_n can be computed directly from the automatically generated interval Taylor coefficients $F_k(C,H)$ and $F_k(X,H)$ of f, $k = 0.1,\ldots,n \leq p$.

Since

$$If = \int_a^b f(x)dx \in J_n, \qquad n = 0,1,\ldots,p, \tag{3.7}$$

the *intersection principle* [8], [9] can be used. One has

$$If \in \bigcap_{n=0}^{p} J_n. \tag{3.8}$$

e interval-valued. One strategy for handling interval-valued endpoints is to allow the ıcertainties in the locations of the endpoints to be carried over into uncertainties in the cations of the nodes, as in INTE [9]. For example, using a one-panel Simpson's rule on decimal digit machine gives

$$\int_{[0,0.1]}^{[3.1,3.2]} f(x)dx \in \frac{(B-A)}{6}\left(F([0,0.1]+4F([1.55,1.65])+F([3.1,3.2])\right)$$

).7)

$$-\frac{(B-A)^5}{2880}F^{iv}([0,3.2]),$$

'here $A = [0,0.1]$, $B = [3.1,3.2]$. For $f(x) = \sin x$, (6.7) yields [1.880,2.214] using the ccurate scalar product, while the correct answer is [1.994,2.000].

A better strategy is to concentrate the uncertainty at the ends:

$$\int_{[0,0.1]}^{[3.1],[3.2]} f(x)dx = \int_{[0,0.1]}^{0.1} f(x)dx + \int_{0.1}^{3.1} f(x)dx + \int_{3.1}^{[3.1,3.2]} f(x)dx. \tag{6.8}$$

'he middle part is handled as an RR integral as described in §§3-4, while the other two ıtegrals, of the types IR and RI, respectively, will be treated in the manner to be described elow. In this example, we can get [1.984,2.073]. As the widths of the endpoints increase, he advantage of the second strategy becomes more pronounced.

The general case (6.3) can be expressed in terms of integrals of the above types. This ase in turn splits into six subcases, according as A and B are (i) disjoint, (ii) overlapping, ır (iii) one is contained in the other. More precisely, for $A = [A_L, A_R]$, $B = [B_L, B_R]$, uppose that $A_R \leq B_R$. Then we have:

Case 1. $AR \leq BL$ (disjoint interval endpoints). Here, the integral (6.3) can be written

$$\int_A^B f(x)dx = \int_{[A_L,A_R]}^{A_R} f(x)dx + \int_{A_R}^{B_L} f(x)dx + \int_{B_L}^{[B_L,B_R]} f(x)dx, \tag{6.9}$$

ınd thus is the sum of integrals of types IR, RR, and RI, respectively.

Case 2. $AL \leq BL \leq AR \leq BR$ (overlapping interval endpoints). Here,

$$\int_A^B f(x)dx = \int_{[A_L,B_L]}^{B_L} f(x)dx + \int_{B_L,A_L}^{B_L,A_L} f(x)dx + \int_{A_R}^{[A_R,B_L]} f(x)dx, \tag{6.10}$$

ıe natural interval inclusions $F_k(X, C_1, \ldots, C_m)$ of f and its Taylor coefficients on an erval $X \in \mathbf{IS}$ are again obtainable on a computer by using interval computation and tomatic differentiation. In particular, for an interval polynomial (6.2), $F_p(X, H) \equiv [0,0]$ $p \geq m$, just as in the real case. The definition

4) $$\int_A^B f(x; C_1, \ldots, C_m)dx = \left\{ \int_A^B f(x, c_1, \ldots, c_m)dx \;\middle|\; c_1 \in C_1, \ldots, c_m \in C_m \right\}$$

cribes the type of integrals to which the methods of this paper apply. It is assumed that B and the coefficient intervals C_i all belong to **IS**, and hence are machine-representable. ıecessary, outward rounding can be used to obtain them from real intervals, preserving lusion of the desired integral.

On the basis of (2.3), it is to be expected that the best results will be obtained for grands f which are very smooth as functions of x. However, one can always compute *interval Riemann sum* $F(X) \cdot w(X)$. This is self-validating, because

) $$\int_X f(x)dx \subseteq F(X) \cdot w(X),$$

is inaccurate and slow to converge [4], [20]. It is important to be able to confine bad avior of the integrand to very small intervals for (6.5) to be useful.

The meaning of tolerance for problems with interval-valued endpoints requires some ification. Consider the problem

) $$\int_0^{[0,1]} \frac{dx}{1+x^2} = \left[\int_0^0 \frac{dx}{1+x^2}, \int_0^1 \frac{dx}{1+x^2} \right] = [0, \tfrac{\pi}{4}].$$

e compute $J = [-0.004, 0.79]$, for example, then $w(J) = 0.794$, although the estimate ach endpoint is in error by less than 0.005. Hence, a requested tolerance must be large gh to accomodate the uncertainties which are inherent in the problem being solved.

The cases that one or both endpoints of integration are nondegenerate intervals in ill now be considered. The possible situations will be denoted by IR, RI, and II, ctively, according as the lower, upper, or both endpoints of the interval of integration

6. Extension to Interval Values

The definition of the integral (1.1) has been extended to arbitrary interval-valued integrands f [4], [20]. For smooth, real-valued functions such as those considered in §2, the interval integral and the Riemann integral coincide. The concept of interval integration is useful in connection with integrands which have jump discontinuities or singularities of various kinds. The definition given in [4] can be extended to the case that the limits of integration are interval-valued:

$$\int_A^B f(x)dx = \left\{ \int_a^b f(x)dx \;\middle|\; a \in A, b \in B \right\}, \tag{6.1}$$

for $A, B \in \mathbf{IR}$, the set of all finite intervals with real endpoints.

There are several reasons to want to be able handle interval endpoints of integration. First of all, some real numbers, such as π, cannot be represented exactly in $\mathbf{S}$, and have to be replaced by the corresponding small intervals such as $[\nabla\pi, \Delta\pi]$ in $\mathbf{IS}$. Secondly, the limits of integration may come from measurements other estimates, and thus are known to lie between certain limits even though their exact values are uncertain. Finally, one can be interested in bounds for the value of the integral (1.1) over ranges of values of limits of integration. In this case, rather than satisfy an accuracy criterion such as (1.2), it is usually desired to find an interval J containing (6.1) which is as small as possible.

Similar considerations apply to interval-valued integrands. The case which usually arises in practice is that the integrand f is a function not only of the independent variable x, but also several parameters $c_1, c_2, \ldots, c_m$. For example, f could be a polynomial of degree $m-1$ with coefficients determined by observations,

$$f(x) = C_1 + C_2 x + \cdots + C_m x^{m-1}, \qquad C_i \in \mathbf{IS},\ i = 1, 2, \ldots, m. \tag{6.2}$$

In general, given intervals $C_1, C_2, \ldots, C_m$, it is natural to define

$$f(x; C_1, \ldots, C_m) = \{ f(x; c_1, \ldots, c_m) \mid c_1 \in C_1, \ldots, c_m \in C_m \}. \tag{6.3}$$

then the integrand cannot be handled by this method. and a message to that effect is sent to the user.

These methods for recognizing and handling certain singularities extend the domain of applicability of the program to include many integrands which arise in applications. However, their usefulness is somewhat limited. For one thing, the location of the singularity must be known in advance, or guessed. For popular sets of test problems, guessing one or both endpoints usually works. For real problems, locations of singularities may be unknown. However, the method given here validates correct guesses. The endpoints to be investigated for the presence of singularities must be machine numbers, not intervals. If an interval-valued endpoint is even one machine number wide, then the problem contains integrands which are unbounded on a set of positive measure. If a singularity is in the interior of the interval of integration, then we can determine its location to within one or two machine numbers. From this, its contribution to If could be estimated, but the possibility that the integrand is unbounded on a set of positive measure cannot be eliminated. Since a validated answer cannot be produced in this case, an error return is selected. If the user knows that the singularity occurs at a machine number, then the integral should be calculated as the sum of two integrals, with the singularity at one endpoint of each.

Suppose that the integrand has the form

$$f(x) = (a - x)^{-s}\phi(x), \tag{5.9}$$

where $\phi(x)$ is analytic at $x = a$. If c is chosen closer to a than any other singularity, then the series for f expanded at $x = a$ is asymptotic to the series for $v(x) = (a - x)^{-s}$. The Taylor coefficients $v_i = v_i(a, h)$ of v satisfy the recurrence relation

$$v_{i+1} = v_i\Big(1 + \frac{s-1}{i}\Big)\frac{h}{R_c},$$

$$s = \Big(\frac{v_{i+1}}{v_i}\frac{h}{R_c} - 1\Big)i + 1, \tag{5.10}$$

where R_c is the radius of convergence of the series. If f cannot be evaluated on $[a, b]$, then one attempts to find constants K_L, K_R, s_L, s_R such that

$$K_L(a - x)^{-s_L} \le f(x) \le K_R(a - x)^{-s_R}, \tag{5.11}$$

for x near a. If such constants can be found and (5.11) validated, then we proceed. If $s_L > 1$, then the program can guarantee that If does not exist. We know of no other numerical quadrature routine which can *validate* nonexistence of an integral. If $s_R < 1$, then

$$\frac{K_L}{1 - s_L}(a - a')^{1-s_L} \le \int_a^{a'} f(x)dx \le \frac{K_R}{1 - s_R}(a - a')^{1-s_R}. \tag{5.12}$$

These bounds can be made as tight as desired by taking a' close enough to a. The interval $[a, a']$ is placed on the list of subintervals, and processing continues on the subinterval $[a', b]$.

A singularity at b can be handled similarly.

If K_L, K_R, s_L, s_R cannot be found, or if

$$s_L < 1 \le s_R, \tag{5.13}$$

because of its width. All other subintervals can be processed using higher-order rules. This strategy applies to singularities in f' which occur anywhere in the interval of integration, and not just at endpoints. It also works for singularities in higher derivatives, whose presence might not even be known to the user. Once a singularity of this type has been confined to a sufficiently small subinterval, it is possible to meet reasonable requirements for accuracy, with self-validation.

5.2. Jump discontinuities

If the integrand is given by an ordinary mathematical expression, then it is difficult to represent a function which has jump discontinuities, and yet can be evaluated at every point in the interval of integration. However, the user can supply a subroutine for evaluation of the integrand which produces jump discontinuities. These appear to the program as singularities in f'. On any subinterval which contains a jump, only the Riemann sum is available, and its width will lead to frequent selection of this subinterval for further processing, as before. No special algorithms are needed in this case.

This behavior is similar to CADRE [b2]. Upon recognizing a jump, CADRE subdivides the interval and uses a low-order rule.

5.3. Some removable singularities

The Taylor series method permits handling of some removable singularities. Consider the problem

$$If = \int_0^\pi \frac{\sin x}{x} dx, \tag{5.8}$$

in which the integrand has a removable singularity at $x = 0$. In this case, the integrand cannot be evaluated directly on any interval containing 0, but f can be expanded at $x = 0$ using l'Hôspital's rule, which can be applied automatically [6]. A short Taylor series with remainder for f at 0 is sufficient to bound If near 0, and the rest of the interval of integration is processed in the normal manner.

5.4. Some algebraic singularities at endpoints

Although the original problem (5.2) is well-behaved, the interval integral in (5.3) contains integrals such as

$$\int_0^{\Delta\pi} \sqrt{\nabla\pi - x}\, dx, \tag{5.4}$$

and consequently does not exist. Hence, the correct response is that (5.3) cannot be evaluated on the entire interval of integration, just on $[0, \nabla\pi]$, for example. This suggests that standard numerical quadrature routines, which avoid evaluation of the integrand at endpoints, can be fooled into returning values for integrals such as

$$\int_0^{\pi} \sqrt{\pi - \epsilon - x}\, dx \tag{5.5}$$

for ϵ sufficiently small, instead of informing the user of possible difficulty. The types of singularities which can be handled by the techniques presented here will now be described.

5.1. Singularities in derivatives of f.

The integrand f may not have enough derivatives for some rules which the integration program can apply. In the process of automatic generation of interval Taylor coefficients on an interval X, the nonexistence of a derivatives of f beyond a certain order is detected and reported. As long as f itself can be evaluated on the entire interval of integration, the order adaptation strategy handles singularities in the derivatives of f. For example, consider the problem

$$If = \int_0^4 \sqrt{x} dx = \frac{4}{3}. \tag{5.6}$$

The first derivative f' is undefined at $x = 0$, so the only rule that can be applied to the entire interval of integration is the Riemann sum

$$\int_0^4 \sqrt{x} dx \in \left[\min_{[0,4]} \sqrt{x}, \max_{[0,4]} \sqrt{x}\right] = [0, 8]. \tag{5.7}$$

At this point the subinterval adaptive strategy takes over. At each step, the subinterval containing 0 requires a Riemann sum, and thus is frequently selected for further processing

5. Treatment of Singularities

Among the quadrature routines which provide estimates for If, the more successful ones have special provisions to handle and perhaps recognize certain types of singularities. QUADPACK [17], for example, can handle integrals of the form

$$\int_a^b (x-a)^\alpha (b-x)^\beta v(x) f(x)\,dx, \tag{5.1}$$

where $\alpha, \beta > -1$ and $v(x) = 1$, $\log(x-a)$, $\log(b-x)$, or $\log(x-a)\log(b-x)$, *provided* the *user* supplies α, β, and the form of v. CADRE [3] attempts to detect and verify the presence of jump discontinuities or x^α-type singularities in the integrand. The program described here attempts to recognize and handle automatically

(1) singularities in derivatives of f,

(2) some removable singularities,

(3) jump discontinuities, and

(4) some algebraic singularities at endpoints.

Since guaranteed bounds are computed for If, the task of such a program is more difficult than for methods which yield only an estimate. For example, Gaussian quadrature can be used in the neighborhood of an endpoint singularity, because the integrand need not be evaluated at the endpoint. To give guaranteed bounds, however, requires that the function, or some of its derivatives, be evaluated on the entire interval. Hence, the price of the guarantee is a restriction on the applicability of the program. Either it verifies that the integrand is in its domain of applicability, or, if it cannot deliver guaranteed bounds, it notifies the user with an indication of the difficulty.

An example will indicate what can go wrong at endpoints. For example,

$$I_1 f = \int_0^\pi \sqrt{\pi - x}\; dx = \frac{2}{3}\pi^{\frac{3}{2}}, \tag{5.2}$$

could be presented to the computer as

$$I_2 f = \int_0^{|\nabla\pi, \Delta\pi|} \sqrt{|\nabla\pi, \Delta\pi| - x}\; dx. \tag{5.3}$$

Because of the difference between the remainder terms in (3.6) for odd and even n, it is prudent to calculate at least one extra value of J_n. However, the conditions above guarantee that no more than two series terms will be computed beyond the one which gives the narrowest I_n.

Another important strategy for order adaptation involves the case that the integrand has singularities in a certain derivative in the given subinterval. This will be discussed more fully in the next section. If a certain derivative cannot be evaluated, this will be detected, and the method will be restricted to rules or orders of Taylor expansion which use only the derivatives which can be evaluated.

The strategy for subinterval adaptation retains all subintervals. At each step, the subinterval which makes the largest contribution to the width of J is processed by breaking it into further subintervals. This processing continues until

(i) $w(J)$ is small enough to satisfy the accuracy requirement (1.2),

(ii) the noise inherent in function evaluation limits further reduction of $w(J)$, or

(iii) more than the maximum number M of function evaluations have been performed.

The second termination criterion in this list is particularly important. If the noise in the function evaluation is large relative to the accuracy requested, eventually the width of the truncation error Ef is made so small that it adds nothing to the width of the rule Rf alone. At this point, further increase in accuracy is not possible. Without the guaranteed bounds provided by the interval computation, many standard methods cannot recognize when this point has been reached, and the calculation should be terminated. Malcolm and Simpson [13] observe that the strategy of processing the worst subinterval results in local errors of roughly equal magnitude. Rice [23] calls this an ordered list interval collection management component, and lists some advantages and disadvantages which will be discussed in more detail in §7.

Alternatively, the actual approximate integrals J_i could be examined, but this requires more computation than use of the error terms alone. Suppose that μ denotes the maximum value of the width $w(F(C))$ of the interval evaluation of f at a node C of the integration formula being used. The total cost can be taken to be proportional to $n_i \cdot w(J_i)$, where n_i is the number of function evaluations. The width $w(J_i)$ can be estimated by $\mu + w(E_{n_i} f)$, and thus i can be chosen as the minimizer of

$$m(i) = \min\{w(J_0), n_i(\mu + w(E_{n_i} f)\}. \tag{4.2}$$

Thus, more nodes are used in a given subinterval only if a significant reduction in width of the approximate integral results. It should also be noted that a given integration rule can be used in several integration formulas having remainder terms of different orders. For example, Stroud and Secrest [24] give error terms for Gaussian integration rules on n nodes which have orders $1 \leq p \leq 2n$. In certain cases, the error terms corresponding to smaller than maximum p can be narrower (for example, for highly oscillatory integrands), and the use of these formulas instead of the standard ones will minimize $w(J_n)$.

In the case of Taylor series, the intersection (3.9) procedure can result in approximations I_n which are considerably better than J_n given by (3.6), which can be viewed as the sum of a rule and an error term. The intervals I_n are monotone decreasing in width, so the increase in accuracy has to be balanced against the cost in time (2.9) of generating more terms of the series. The constants a, b, c in (2.9) can always be determined for a given integrand, so the corresponding heuristic can be based on the function

$$\theta(n) = w(I_n) \cdot (an^2 + bn + c). \tag{4.3}$$

Generation of the Taylor series can be stopped when

(i) $I_{n-2} = I_{n-1} = I_n$,

(ii) $\theta(n-2) \leq \theta(n)$, or

(iii) $n = p$.

4. Adaptive Strategies

The computer program INTE [9] demonstrated the reality of automatic, self-validating numerical quadrature using Newton-Cotes and Gaussian integration formulas in the way discussed in §2. (Euler-Maclaurin integration was added to the capabilities of INTE later [10].) However, there was no attempt to address the problem of efficiency, a defect remedied in the program described later. In order to satisfy the accuracy criterion (1.2), if possible, INTE simply divided the interval of integration into a sufficient number of equal subintervals [9], [18]. By contrast, popular numerical integration packages such as CADRE [3] and QUADPACK [17] use information obtained about the behavior of the integrand to attempt to reduce the number of function evaluations to a minimum. The method given here uses similar strategies, except that estimates of the error based on evaluation of the integrand at a finite set of points are replaced by guaranteed bounds. This eliminates the need for "safety factors" [23].

Adaptive strategies fall into the categories of *order* adaptation, which relates to the choice of the formula used in each subinterval, and *subinterval* adaptation, which determines how the original interval of integration is broken up into subintervals. Order adaptation is somewhat simpler than subinterval adaptation, and will be considered first. For methods based either on standard quadrature formulas or Taylor series, *order zero* refers to the interval Riemann sum $F(X) \cdot w(X)$, which always contains $\int_X f(x)dx$ [4].

Given a suite of numerical integration formulas of the form (2.3), suppose that $X \in \mathbf{IS}$ is the current subinterval of integration. Specifically, suppose that the given rules are of order i for $i = 0, 1, \ldots, k$ on n_i points. Once the interval Taylor coefficients $F_i(X, H)$ have been formed, then the order $p \leq k$ of the most accurate rule can be chosen to be the value of i for which the width of the error term

$$E_{n_i} f = C_{n_i} H \cdot F_i(X, H) \tag{4.1}$$

is minimum, $i = 0, 1, \ldots, k$.

Alternatively, the actual approximate integrals J_i could be examined, but this requires more computation than use of the error terms alone. Suppose that μ denotes the maximum value of the width $w(F(C))$ of the interval evaluation of f at a node C of the integration formula being used. The total cost can be taken to be proportional to $n_i \cdot w(J_i)$, where n_i is the number of function evaluations. The width $w(J_i)$ can be estimated by $\mu + w(E_{n_i} f)$, and thus i can be chosen as the minimizer of

$$m(i) = \min\{w(J_0), n_i(\mu + w(E_{n_i} f)\}. \tag{4.2}$$

Thus, more nodes are used in a given subinterval only if a significant reduction in width of the approximate integral results. It should also be noted that a given integration rule can be used in several integration formulas having remainder terms of different orders. For example, Stroud and Secrest [24] give error terms for Gaussian integration rules on n nodes which have orders $1 \leq p \leq 2n$. In certain cases, the error terms corresponding to smaller than maximum p can be narrower (for example, for highly oscillatory integrands), and the use of these formulas instead of the standard ones will minimize $w(J_n)$.

In the case of Taylor series, the intersection (3.9) procedure can result in approximations I_n which are considerably better than J_n given by (3.6), which can be viewed as the sum of a rule and an error term. The intervals I_n are monotone decreasing in width, so the increase in accuracy has to be balanced against the cost in time (2.9) of generating more terms of the series. The constants a, b, c in (2.9) can always be determined for a given integrand, so the corresponding heuristic can be based on the function

$$\theta(n) = w(I_n) \cdot (an^2 + bn + c). \tag{4.3}$$

Generation of the Taylor series can be stopped when

(i) $I_{n-2} = I_{n-1} = I_n$,

(ii) $\theta(n-2) \leq \theta(n)$, or

(iii) $n = p$.

4. Adaptive Strategies

The computer program INTE [9] demonstrated the reality of automatic, self-validating numerical quadrature using Newton-Cotes and Gaussian integration formulas in the way discussed in §2. (Euler-Maclaurin integration was added to the capabilities of INTE later [10].) However, there was no attempt to address the problem of efficiency, a defect remedied in the program described later. In order to satisfy the accuracy criterion (1.2), if possible, INTE simply divided the interval of integration into a sufficient number of equal subintervals [9], [18]. By contrast, popular numerical integration packages such as CADRE [3] and QUADPACK [17] use information obtained about the behavior of the integrand to attempt to reduce the number of function evaluations to a minimum. The method given here uses similar strategies, except that estimates of the error based on evaluation of the integrand at a finite set of points are replaced by guaranteed bounds. This eliminates the need for "safety factors" [23].

Adaptive strategies fall into the categories of *order* adaptation, which relates to the choice of the formula used in each subinterval, and *subinterval* adaptation, which determines how the original interval of integration is broken up into subintervals. Order adaptation is somewhat simpler than subinterval adaptation, and will be considered first. For methods based either on standard quadrature formulas or Taylor series, *order zero* refers to the interval Riemann sum $F(X) \cdot w(X)$, which always contains $\int_X f(x)dx$ [4].

Given a suite of numerical integration formulas of the form (2.3), suppose that $X \in \mathbf{IS}$ is the current subinterval of integration. Specifically, suppose that the given rules are of order i for $i = 0, 1, \ldots, k$ on n_i points. Once the interval Taylor coefficients $F_i(X, H)$ have been formed, then the order $p \leq k$ of the most accurate rule can be chosen to be the value of i for which the width of the error term

$$E_{n_i} f = C_{n_i} H \cdot F_i(X, H) \tag{4.1}$$

is minimum, $i = 0, 1, \ldots, k$.

This intersection can be calculated as the corresponding interval Taylor coefficients of the integrand are generated:

$$(3.9)\qquad \begin{cases} I_0 = J_0, & \text{(a Riemann sum)}, \\ I_n = I_{n-1} \bigcap J_n, & n = 1, 2, \ldots, p. \end{cases}$$

This provides a means to determine the highest useful term of the Taylor expansion, since the calculation can be terminated when effective decrease in the widths of the intervals $\{I_n\}$ ceases, or when the desired tolerance is met.

As in the case of (2.12), formula (3.6) can be evaluated as the scalar product of the vectors

$$(3.10)\qquad \mathbf{F} = \big(F_0(C,H), F_1(C,H), \ldots, F_{n-1}(C,H), T_n(C,H)\big),$$

where $T_n(C,H) = F_n(C,H) - F_n(C,H)$ or $T_n(C,H) = 2F_n(C,H)$ according as n is odd or even, and the vector

$$(3.11)\qquad \mathbf{W} = \big(W_0, W_1, \ldots, W_{n-1}, W_n\big),$$

where

$$(3.12)\qquad W_k = \frac{H}{k+1}, \qquad k = 0, 1, \ldots, n.$$

The width of J_n can also be reduced somewhat if the expansion is about $C = [c,c]$, $c \in \mathbf{S}$. In this case, the interval stepsize H might have to be increased a very small amount to maintain inclusion of the subinterval of integration. In §5, it will be noted that expansion about an endpoint, as in (3.1), or some other point in the interval of integration, can be helpful if the integrand has removable singularities.

the sum of integrals of types IR, II, and RI.

Case 3. $BL < AL \leq AR \leq BR$ (A is properly contained in B). Here,

$$\int_A^B f(x)dx = -\int_{[B_L,A_L]}^{A_L} f(x)dx + \int_{[A_L,A_R]}^{[A_L,A_R]} f(x)dx + \int_{A_R}^{[A_R,B_R]} f(x)dx, \tag{6.11}$$

again the sum of integrals of types IR, II, and RI.

The other three cases are obtained for $B_R < A_R$ by reversing the rôles of A and B in the preceding.

Case 1 is undoubtedly the one which is encountered most often in practice. To illustrate Case 3, consider

$$\int_{[2,3]}^{[0,5]} f(x)dx = -\int_{[0,2]}^{2} f(x)dx + \int_{[2,3]}^{[2,3]} f(x)dx + \int_{3}^{[3,5]} f(x)dx. \tag{6.12}$$

We are not aware of physical problems which give rise to integration problems other than Case 1 (except to bound the values of integrals over ranges of endpoints), but provision for these cases adds little to the machinery which is required to handle Case 1.

Integrals of types RI, IR, and II can be handled by Gauss or Newton-Cotes formulas with interval-valued nodes, but this leads to wide interval bounds. The method of Taylor series, as outlined in §4, applies as easily to the cases of one or both endpoints interval-valued as to RR type integration. Hence, the program uses Taylor polynomials for integrals of types RI, IR, and II, even if the user has chosen Gauss or Newton-Cotes formulas for use on type RR subintervals.

Using the same notation as in §3, $g(x)$ denotes an indefinite integral (3.4) of $f(x)$. The one-panel form of the various integration formulas are then:

Type RI:

$$\int_a^{[a,b]} f(x)dx \subseteq g(x)\Big|_a^{[a,b]} + F^{(n)}([a,b])\frac{(x-c)^{n+1}}{(n+1)!}\Big|_a^{[a,b]}, \tag{6.13}$$

Type IR:

$$\int_{[a,b]}^{b} f(x)dx \subseteq g(x)\Big|_{[a,b]}^{b} + F^{(n)}([a,b])\frac{(x-c)^{n+1}}{(n+1)!}\Big|_{[a,b]}^{b}. \tag{6.14}$$

Type II:

$$\int_{[a,b]}^{[a,b]} f(x)dx \subseteq g(x)\Big|_{[a,b]}^{[a,b]} + F^{(n)}([a,b])\frac{(x-c)^{n+1}}{(n+1)!}\Big|_{[a,b]}^{[a,b]}. \tag{6.15}$$

Each uses intersections of subsequent estimates and order adaptation as the RR algorithm does.

Formulation of the multipanel forms for the above types requires some care. Suppose the nodes $a = x_0 < x_1 < \cdots < x_k = b$ are selected, and set $Y_i = [x_{i-1}, x_i]$ for $i = 1, 2, \ldots, k$. For type II integrals, let

$$I_{Y_i} f = \int_{Y_i}^{Y_i} f(x)dx, \tag{6.16}$$

which is symmetric about 0. If $s, t \in Y_i$, then

$$\int_s^t f(x)dx \in I_{Y_i} f \subseteq \sum_{i=1}^{k} I_{Y_i} f. \tag{6.17}$$

If $s \in Y_i$ and $t \in Y_j$ with $i < j$, then

$$\begin{aligned} \int_s^t f(x)dx &= \int_s^{x_i} f(x)dx + \int_{x_i}^{x_{i+1}} f(x)dx + \cdots + \int_{x_{j-1}}^{t} f(x)dx \\ &\in \int_{Y_i}^{Y_i} f(x)dx + \int_{Y_{i+1}}^{Y_{i+1}} f(x)dx + \cdots + \int_{Y_j}^{Y_j} f(x)dx \\ &\subseteq \sum_{i=1}^{k} I_{Y_i} f. \end{aligned} \tag{6.18}$$

Hence,

$$\int_{[a,b]}^{[a,b]} f(x)dx \subseteq \sum_{i=1}^{k} \int_{Y_i}^{Y_i} f(x)dx. \tag{6.19}$$

The subintervals Y_i are chosen by the same subinterval adaptive strategy as used for type RR integrals. Thus, the algorithm for type II integrals is the same as for type RR, except that the integral on each subinterval is computed using the one-panel type II rule, equation (6.15), with intersection.

The multipanel rules for types RI and IR do not lend themselves to subinterval adaptation. By definition,

$$\begin{aligned}\int_a^{[a,b]} f(x)dx &= \left\{ \int_a^t f(x)dx \;\middle|\; t \in [a,b] \right\} \\ &= \bigcup_{i=1}^{k} \left\{ \int_a^t f(x)dx \;\middle|\; t \in Y_i \right\} \\ &= \int_a^{Y_1} f(x)dx + \\ &\quad + \int_a^{x_1} f(x)dx + \int_{x_1}^{Y_2} f(x)dx + \\ &\quad + \cdots + \\ &\quad + \int_a^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{k-1}}^{Y_k} f(x)dx\end{aligned} \tag{6.20}$$

Once the endpoints of a subinterval are chosen, that subinterval can be subdivided only with great difficulty. We observe that

$$\int_a^{[a,b]} f(x)dx \subseteq \int_{[a,b]}^{[a,b]} f(x)dx, \tag{6.21}$$

so the type II algorithm gives bounds using adaptation. If those bounds are not small enough, then we apply the algorithm given by equation (6.20) using a fixed stepsize equal to the smallest stepsize chosen adaptively by the type II algorithm.

Type IR integrals are done similarly.

In the general case of the integral (6.3), the allowable tolerances and the maximum number of function evaluations must be apportioned among three distinct, independent integrations. These are given in proportion to the width of the respective subintervals. If $\tau(X_i)$ denotes the tolerance allowed on the subinterval X_i, τ^* is the tolerance remaining, then

$$\tau(X_i) = \tau^* \cdot \frac{w(X_i)}{w(X)}. \tag{6.21}$$

If the integral on the first subinterval is narrower than its share of the total tolerance requires, then the tolerances on the other subintervals are relaxed so that the total tolerance

can be met more efficiently. On the other hand, if the integral on the first subinterval is a little too wide, then the integrals on the remaining subintervals can sometimes be commuted accurately enough that the total tolerance can still be satisfied.

7. Implementation Details

It follows from the above discussion that realization of adaptive, self-validating quadrature routines on a computer requires that the following features be supported:

(i) Interval computation (arithmetic operations and standard library functions).

(ii) Subroutines for automatic generation of Taylor coefficients.

(iii) Accurate scalar product of interval vectors (to minimize width due to roundoff error).

The program INTE [9] was written in FORTRAN for the Sperry 1100, on which only (i) [25] and (ii) were supported. As mentioned above, INTE performs self-validating numerical integration without adaptation. The microcomputer language Pascal-SC [22] and the ACRITH package for the IBM 370 series of computers support (i) and (iii), to which the routines (ii) have been added. The implementation described in [6] for Pascal-SC is immediately adaptable to ACRITH. The program described here is written in FORTRAN for use with ACRITH. In particular, the following operators and library functions are supported:

+	SIN	SINH	EXP
−	COS	COSH	LOG=LN
*	TAN	TANH	LOG10
/	COTAN=COT	COTANH=COTH	ERF
** constant	ASIN	ASINH	ERFC
ABS	ACOS	ACOSH	GAMMA
SQR	ATAN	ATANH	LGAMMA
SQRT	ACOTAN=ACOT	ACOTNH=ACOTH	

The Taylor series terms $F_n(C,H)$ and $F_n(X,H)$ are calculated by using the well-known recurrence relations [15], [16], [19] for the operators and functions given above. In this application, the "point" of expansion and the stepsize are interval-valued to give the desired inclusions, but the recurrence relations remain the same. It is possible for the user to augment the list of library functions given above if the required recurrence relation for Taylor coefficients of the new function is known.

Given an integrand of the type considered, for example,

$$f(x) = \frac{4}{1+x^2}, \tag{7.1}$$

the program first parses it into a *code list* [19]:

Operator	Operand 1	Operand 2	Result
SQR	x (=Temp1)		Temp4
+	1 (=Temp2)	Temp4	Temp5
/	4 (=Temp3)	Temp5	F

In order to compute the series for f expanded at C with stepsize H, this code list is interpreted to obtain the sequence of calls

```
Temp1 := (C,H) {The series for x.}
Temp2 := (1)   {Constant series.}
Temp3 := (4)   {Constant series.}
Call ITSQR(Temp1,Temp4)
Call ITADD(Temp2,Temp4,Temp5)
Call ITDIV(Temp3,Temp5,F).
```

The result is an array which contains the interval-valued series for f and an indication of how many terms were computed. For example, the subroutine ITSQR(U,V) computes the

series for $V = U^2$, given the series for U, by means of the recurrence relations

$$V_i = \begin{cases} 2 * \sum_{j=1}^{\frac{i-1}{2}} U_j * U_{i-j+1} + \mathrm{ISQR}\left(U_{\frac{i+1}{2}}\right), & i \text{ odd}, \\ 2 * \sum_{j=1}^{i/2} U_j * U_{i-j+1}, & i \text{ even}, \end{cases} \tag{7.2}$$

where $V_i = V_i(C, H)$ and $U_i = U_i(C, H)$ denote the ith Taylor coefficents of U and V, respectively [19], p. 49. The interval function ISQR computes $\mathrm{ISQR}(X) = X^2$ instead of $X \cdot X$, which is preferable in general, since, for example, $[-1,1]^2 = [0,1]$ while $[-1,1] \cdot [-1,1] = [-1,1]$. The parsing and interpretation at runtime described above is unnecessary in Pascal-SC, because the compiler generates the required code [6].

The interval Taylor coefficients $F_1(X, C)$, $F_2(X, C)$, ... are maintained in a record-like structure. If "F" is the name of the function being expanded, then

LF	Index of last known nonzero term;
MF	Index of last known term;
OFL	Vector of series terms—left (lower) bound;
OFR	Vector of series terms—right (upper) bound.

The designations for other variables replace "F" in the above.

The algorithms used depend on whether the endpoints of the interval X of integration are elements of **S**, that is, machine numbers, or whether they are intervals. The basic algorithm is for the case $X = [a, b] \in \mathbf{IS}$, and is called the RR (REAL-REAL) algorithm:

1. Compute the integral on $[a, b]$;
2. Add $[a, b]$ to the list of subintervals;
3. Loop
    4. Find the subinterval on which the width of the integral is largest;
    5. Bisect it;
    6. Compute the integral on the left subinterval;

7. Add the left subinterval to the list;
8. Compute the integral on the right subinterval;
9. Add the right subinterval to the list;
10. Compute the integral on $[a, b]$ by summing the integrals on all the subintervals;
11. Exit when accuracy tolerance is met;
12. Exit with warning when
13. no further improvement in accuracy is possible,
14. or M function evaluations are exceeded;
15. End loop.

Each subinterval X is maintained in a data structure of the following form:

XA	Left endpoint of the subinterval;
XB	Right endpoint of the subinterval;
OPTORD	Order of the derivative used to compute the remainder;
WIDINT	Width of the integral on this subinterval;
SINT	Interval-valued integral on this subinterval;
WEGHT	Vector of interval valued weights (Gauss and Newton-Cotes), stepsize (Taylor);
FNVAL	Vector of interval-valued function values (Gauss and Newton-Cotes), series terms (Taylor);
FNTRN	Vector of interval-valued function values (Gauss and Newton-Cotes), series terms (Taylor), including remainder terms.

At steps 1, 6, and 8, the integral is computed on the subinterval [XA,XB] using one-panel versions of Gauss, Newton-Cotes, or Taylor polynomials as outlined in Sections 2 and

3. The weight vectors and functions are arranged in the vectors WEGHT and FNTRN, respectively, in such a way that the interval inclusion

$$J = \sum_i \mathrm{WEGHT}_i * \mathrm{FNTRN}_i \tag{7.3}$$

of $\int_a^b f(x)dx$ is computed as a single inner product. Similarly,

$$Rf = \sum_i \mathrm{WEGHT}_i * \mathrm{FNVAL}_i. \tag{7.4}$$

At step 13, if $J \subseteq Rf$, then the loop is exited. In this case, further reduction of the width of the truncation error cannot reduce $w(J)$. For Gauss and Newton-Cotes formulas, SINT is used only to give WIDINT. For Taylor polynomials, $\sum_i \mathrm{SINT}_i$ is intersected with (7.3). SINT_i is computed using the intersection principle discussed at the end of §3. For a few subintervals, $\sum_i \mathrm{SINT}_i$ is narrower than (7.3), while the situation is usually reversed for a large number of subintervals, because (7.3) uses the accurate scalar product.

The arrangement of subintervals in the arrays listed above must be relatively straightforward to allow J to be computed by a single scalar product operation. Each iteration of the loop from step 3 to step 15 removes one subinterval from the list and replaces it with two subintervals. Subintervals are not otherwise deleted from the list. Hence, the following simple allocation scheme works: On the ith pass through the loop, the information about the left subinterval is stored in the locations previously used by its parent, and the information about the right subinterval is stored in the $(i + 1)$st locations, following the already computed values. Hence, insertion requires no searching. The widest subinterval is found at step 4 by a sequential search of the array WIDINT(1..i).

By contrast, QUADPACK [17] maintains its list of pending subintervals in sorted order, so no search is necessary for the next subinterval to be processed. However, new subintervals are inserted at locations found by a sequential search, followed by changing pointers to all following entries in the list. For each subinterval processed, the program does one sequential search, while QUADPACK does two. In addition, QUADPACK uses two sets of pointer adjustments.

For integration using Taylor polynomials, the maintenance of list of subintervals is somewhat more complicated, because the program reuses the series which it has previously computed. To illustrate the ideas, consider the first execution of the loop at step 3. At that point, the list of subintervals contains only one: $X = [a, b]$ itself. FNTRN contains OPTORD−1 terms of the series for f expanded at $c = (a + b)/2$ and the truncation error term involving $F^{(\mathrm{OPTORD})}(X)$ given by equation (4.6). Provided that the requested tolerance exceeds the noise inherent in the function evaluation, a stepsize h can be computed which is small enough that the requested tolerance per unit step is satisfied on the interval $[c - h, c + h]$. Notice that if a relative tolerance is requested, then this requires a current estimate for J. The value of the integral on this subinterval can be computed at a cost proportional to OPTORD, instead of a cost proportional to OPTORD^2, which would be required to generate J directly by using the recurrence relations for f. Following this, the two subintervals $[a, c - h]$ and $[c + h, b]$ are processed directly. Consequently, this method breaks the subinterval of integration into three parts, rather than bisecting it.

Thus, for integration by Taylor polynomials, step 5 of the RR algorithm is replaced by

5.0′ Compute h such that the tolerance is satisfied on $[c - h, c + h]$;

5.1′ Compute the integral on $[c - h, c + h]$ from information in FNVAL;

5.2′ "left subinterval" := [XA, $c - h$];

5.3′ "right subinterval" := [$c + h$, XB].

The middle subinterval $[c - h, c + h]$ is maintained on the list of subintervals so that its contribution to J is included in the scalar product in (7.3). This has the helpful side effect that h can be chosen somewhat optimistically. If the choice is too optimistic, then $[c - h, c + h]$ will be selected later for further processing as the worst subinterval, at which time it will be broken into three parts. One of the new subintervals will occupy the place of the parent interval in the list maintained, while the other two will be added to the end of the list.

A further refinement could be implemented. The stepsize h computed at step 5.0′ has the following property: On each subinterval of length $2h$ which is contained in [XA,XB], the use of a Taylor polynomial of degree OPTORD−1 yields an integral which satisfies the requested tolerance. The integration on such a subinterval can be done with half the usual work. No series for the truncation error needs to be computed because the truncation error can be bounded by using the global remainder term on [XA,XB]. If [XA,XB] can be covered by a few subintervals of length $2h$, then this could be done, and division into three parts would be needed only when the middle part is relatively small. This refinement could improve the efficiency of the program. However, the improvement would likely be modest, because the advantages of intersecting subsequent estimates on each small subinterval would be lost, and the number of subintervals added to the list would no longer be constant. The program also does not reuse function evaluations required by Gauss or Newton-Cotes formulas, although it could be modified to do so.

8. Numerical Examples

We give four examples to illustrate the accuracy and the reliability of the program. All computations were done in double precision on an IBM 4341 computer, using calls to ACRITH routines for all necessary interval calculations, including scalar products.

Example 1. $I = \int_{0.6}^{0.7} \frac{1}{1-x}\,dx.$

Interval bounds for the answer can be computed in three ways:

$$I = \log(1 - 0.6) - \log(1 - 0.7), \tag{8.1}$$

$$I = \log(4/3), \tag{8.2}$$

or by adaptive, self-validating quadrature.

Neither 0.6 nor 0.7 are machine numbers, so they are converted to intervals which are one machine number wide. All three methods give the interval [0.2876820724517808,

876820724517811], but the results are 20, 13, and 14 machine numbers wide, respec-
ely. That is, the program is capable of accuracies comparable to evaluation of the
alytic expression for the answer. This result required 43 equivalent function evaluations
18 subintervals. In order to appreciate the accuracy achieved by the program, we must
sider some details at the level of machine numbers. If (8.1) is evaluated without sim-
ication using interval calculations, the result is the hexadecimal interval [Z 4049 A588
)3 6E41, Z 4049 A588 44D3 6E55], which is 20 machine numbers wide. Using (8.2), the
4 hexadecimal digits are [Z ... 6E45, Z ... 6E52], which is 13 machine numbers wide.
adaptive, self-validating quadrature program gives [Z ... 6E44, Z ... 6E52], which is
nachine numbers wide.

Example 2. $\int_0^4 \sqrt{x}dx = \frac{16}{3}$ (see §5.1).

This example illustrates the order adaptation required to handle the nonexistance of
). The program gave the interval [5.33333 33333 27, 5.33333 33333 36]. It stopped when
d used 400 effective function evaluations on 226 subintervals. Most of the subintervals
clustered near the origin. Away from 0, as many as 13 series terms were used.

Example 3. $\int_0^1 f(x)dx$, where $f(x) = \begin{cases} 0, & x < 0.3, \\ 1 & x \geq 0.3. \end{cases}$

This example illustrates integration of a simple discontinuous function. The parser
not accept a piece-wise definition, so this function was coded by hand. Table 1 shows
esults for tolerances 0.0 and 1.0E-15, and for recognizing that the function has a
larity on the interval of integration.

	Width of Integral	Function Evaluations	Subintervals
Tolerance = 0.0			
[0, 1]	2.50E-16	218	146
[0, 0.3] + [0.3, 1]	2.78E-17	22	29
Tolerance = 1.0E-15			
[0, 1]	7.77E-16	194	130
[0, 0.3] + [0.3, 1]	1.39E-16	10	5

Table 1. Integrating a Discontinuous Function.

As is usually the case, the program performs *much* better when the user recognizes the presence of a discontinuity. The performance would be even better if the discontinuity occured at a machine number. Notice that the cost of requesting the program to do the best it can (tolerance = 0.0) is only slightly more than the cost when a lesser tolerance is prescribed. The table shows more subintervals than evaluations because evaluating a series which is $\equiv 0$ is not counted as an evaluation.

Example 4. $\int_0^1 \frac{1}{1-\alpha x^2} dx.$

This example illustrates the performance of the program as the integrand varies from very smooth to being undefined at a point in the interval of integration. These calculations

done in single precision with an absolute error tolerance request of 1.0E-5.

Alpha	Error Code	Function Evaluations	Subintervals	Maximum Order	Absolute Error
0.00	0	2	2	3	0.0
0.05	0	8	2	20	4.4E-16
0.10	0	8	2	20	1.8E-14
0.15	0	8	2	20	6.6E-12
0.20	0	8	2	20	7.8E-10
0.25	0	5	2	15	4.1E-06
0.30	0	15	6	15	1.9E-10
0.35	0	12	6	13	4.5E-08
0.40	0	12	6	13	1.8E-07
0.45	0	21	10	15	6.1E-08
0.50	0	19	10	14	1.4E-07
0.55	0	16	10	12	7.9E-06
0.60	0	24	14	12	2.4E-07
0.65	0	35	18	15	6.4E-08
0.70	0	29	18	12	5.0E-06
0.75	0	38	22	12	1.3E-07
0.80	0	48	26	15	6.6E-08
0.85	0	43	26	12	7.5E-06
0.90	0	62	34	15	6.8E-08
0.95	0	73	42	14	1.2E-07
1.00-1.05	66 67	254 2	150	8	3.2E-01

Table 2. Performance Profile.

An error code of 66 signals that the program was unable to meet the requested tol-
e, while 67 means that it was unable to evaluate the integrand. As the problem
ne more difficult, the program required more effective function evaluations and more
tervals.

References

G. Alefeld and J. Herzberger. Introduction to Interval Computation, tr. by J. Rokne. Academic Press, New York, 1983.

Carl de Boor. On writing an automatic integration algorithm, pp. 201-209 in *Mathematical Software*, ed. by John R. Rice, Academic Press, New York, 1971.

Carl de Boor. An algorithm for numerical quadrature, pp. 417-449 in *Mathematical Software*, ed. by John R. Rice, Academic Press, New York, 1971.

Ole Caprani, Kaj Madsen, and L. B. Rall. Integration of interval functions. *SIAM J. Math. Anal.* **12**, no. 3 (1981), 321-341.

G. F. Corliss and Y. F. Chang. Solving ordinary differential equations using Taylor series. *ACM Trans. Math. Software* **8** (1982), 114-144.

G. F. Corliss and L. B. Rall. Automatic generation of Taylor coefficients in Pascal-SC: Basic applications to ordinary differential equations. *Transactions of the First Army Conference on Applied Mathematics and Computing*, pp. 177-209. U. S. Army Research Office, Research Triangle Park, N. C., 1984.

P. J. Davis and P. Rabinowitz. Methods of Numerical Integration, 2nd ed. Academic Press, New York, 1984.

Julia H. Gray and L. B. Rall. A computational system for numerical integration with rigorous error estimation. *Proceedings of the 1974 Army Numerical Analysis Conference*, pp. 341-355. U. S. Army Research Office, Research Triangle Park, N. C., 1974.

Julia H. Gray and L. B. Rall. INTE: A UNIVAC 1108/1110 program for numerical integration with rigorous error estimation. *MRC Technical Summary Report No. 1428*, University of Wisconsin-Madison, 1975.

Julia H. Gray and L. B. Rall. Automatic Euler-Maclaurin integration. *Proceedings of the 1976 Army Numerical Analysis and Computers Conference*, pp. 431-444. U. S. Army Research Office. Research Triangle Park, N. C., 1976.

11. U. W. Kulisch and W. L. Miranker. Computer Arithmetic in Theory and Practice. Academic Press, New York, 1981.

12. U. W. Kulisch and W. L. Miranker (Eds.). A New Approach to Scientific Computation. Academic Press, New York, 1983

13. M. A. Malcolm and R. B. Simpson. Local vs. global strategies for adaptive quadrature. *ACM Trans. Math. Software* **1**, no. 2 (1975), 127-146.

14. R. E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers, pp. 61-130 in *Error in Digital Computation, Vol. 1*, ed. by L. B. Rall. Wiley, New York, 1965.

15. R. E. Moore. Interval Analysis. Prentice-Hall, Englewood Cliffs, N. J., 1966.

16. R. E. Moore. Techniques and Applications of Interval Analysis. SIAM Studies in Applied Mathematics, 2, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

17. R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner. QUADPACK: A Subroutine Package for Automatic Integration. Springer Series in Computational Mathematics, No. 1. Springer, New York, 1983.

18. L. B. Rall. Optimization of interval computation, pp. 489-498 in *Interval Mathematics 1980*, ed. by K. L. E. Nickel, Academic Press. New York, 1980.

19. L. B. Rall. Automatic Differentiation: Techniques and Applications. Lecture Notes in Computer Science, no. 120. Springer, New York, 1981.

20. L. B. Rall. Integration of interval functions II. The finite case. *SIAM J. Math. Anal.* **13**, no. 4 (1982), 690-697.

21. L. B. Rall. Representations of intervals and optimal error bounds. *Math. of Comp.* **41**, no. 163 (1983), 219-227.

22. L. B. Rall. An introduction to the scientific computing language Pascal-SC. *Transactions of the Second Army Conference on Applied Mathematics and Computing*, pp.

117-148. U. S. Army Research Office, Research Triangle Park, N. C., 1985.

23. J. R. Rice. A metalgorithm for adaptive quadrature. *J. ACM* **22**, no. 1 (1975), 61-82.

24. A. H. Stroud and D. Secrest. Gaussian Quadrature Formulas. Prentice-Hall. Englewood Cliffs, N. J., 1966.

25. J. M. Yohe. The interval arithmetic package. *MRC Technical Summary Report No. 1755*, University of Wisconsin-Madison, 1977.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER #2815	2. GOVT ACCESSION NO. AD-A158164	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADAPTIVE, SELF-VALIDATING NUMERICAL QUADRATURE		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) George F. Corliss and L. B. Rall		8. CONTRACT OR GRANT NUMBER(s) DAAG29-80-C-0041
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit Number 3 - Numerical Analysis and Scientific Computing
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office P.O. Box 12211 Research Triangle Park, North Carolina 27709		12. REPORT DATE May 1985 13. NUMBER OF PAGES 37
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Numerical quadrature, Guaranteed error bounds, Automatic Differentiation, Interval computation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Integrals of a function of a single variable can be expressed as the sum of a numerical quadrature rule and a remainder term. The quadrature rule is a linear combination of function values and weights, or the integral of a Taylor polynomial, while the remainder term depends on some derivative of the integrand evaluated at an unknown point in the interval of integration. Numeric[illegible] quadrature is made self-validating by using interval computation to capture both the roundoff and truncation errors made when using a given rule. Necessary derivatives can be generated automatically by using well-known recurrence relations

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. for Taylor coefficients. In order for quadrature methods of this type to be accurate (in the sense that small intervals containing the exact result are produced) and efficient (to obtain results of given accuracy in a reasonably short time), an accurate scalar product and an adaptive strategy are required. The necessary scalar product and support for interval arithmetic are provided in Pascal-SC (for microcomputers) and ACRITH (for IBM 370 computers). The adaptive strategy chooses the subintervals of integration and the order of the quadrature formula used in each subinterval on the basis of guaranteed, rather than estimated, information about the error of the numerical integration in each subinterval. The program described in this report implements standard Newton-Cotes, Gaussian, and Taylor series methods for numerical integration. Ways to handle singularities are discussed, and comparisons are given with a standard numerical integration method.

END

FILMED

9-85

DTIC