Numerical Analysis

J. F. TRAUB, Editor

Automatic Integration of a Function with a Parameter

PHILIP RABINOWITZ*
Brown University,† Providence, Rhode Island

Two efficient methods for automatic numerical integration are Romberg integration and adaptive Simpson integration. For integrands of the form $f(x)g(x, \alpha)$ where α is a parameter, it is shown that Romberg's method is more efficient. A FORTRAN program shows how to achieve this greater efficiency.

The use of automatic integration schemes has proven very attractive to anyone who has had to evaluate a definite integral on a computer. The user need only prepare a program for the integrand and specify the limits of integration and a tolerance ϵ to be reasonably certain that the automatic integration program will compute a value for his integral that is correct to within the given tolerance ϵ. If his tolerance is too small, the program will inform him of this and give him some information which will enable him to decide what to do next, such as subdividing the original interval of integration or accepting a larger error. Of course, it is easy to fix up examples which will "beat" any of these schemes. However, since these almost never occur in practice, we can safely use these schemes.

There are two principal methods of automatic integration which have proved themselves in practice to be both accurate and efficient in terms of the number of functional evaluations needed to achieve a specified accuracy. One method is Romberg integration, which has been the subject of a series of investigations by Bauer, Stiefel, Rutishauser and others [2, 3, 5–7, 10, 12, 13, 20–26]. Programs exist both in ALGOL [1, 2, 4–6, 11, 14] and in FORTRAN [8]. The other method of automatic integration is adaptive Newton-Cotes integration, which has been brought to the attention of the public only in the form of algorithms published in the Algorithms Section of the *Communications of the ACM* [15, 16, 18, 19]. Since the general Newton-Cotes algorithm was given in recursive form [19] and whereas the adaptive Simpson's rule integration was also given in nonrecursive form [18] suitable for translating into a FORTRAN program, our discussion is limited to the latter. There has also been an attempt to combine Romberg integration with an adaptive scheme [9, 21]. However this method requires more information to be given by the user and appears to be less efficient than the other schemes.

A principal difference between Romberg integration and adaptive integration is that the former is a global scheme and the sequence of evaluation points for the integrand is independent of the nature of the integrand although, of course, the number of such points will depend very much on its form. On the other hand, the adaptive scheme is local, and hence, both the sequence and the number of evaluation points depend on the nature of the integrand.

If nothing is known about the integrand, adaptive integration is preferred to Romberg integration because of its ability to handle more efficiently functions which have singularities themselves or in their derivatives [27]. In fact, the adaptive Simpson's rule integration routine is so good that it even integrated correctly over the singularity in an integrand whose integral is convergent [17]. It is not the purpose of this paper to discuss the relative merits of these two integration schemes in general and we will limit ourselves to listing some typical examples of Romberg and adaptive Simpson integration in Table I.

There is one situation where the use of Romberg integration presents a considerable saving in computation time. This is the case where the integrand is of the form $h(x, \alpha) = f(x)g(x, \alpha)$ where α is a parameter, and interest centers in the integral for a series of values of α. More generally, $h(x, \alpha)$ may contain several subexpressions which do not depend on α. However, the case $h(x, \alpha) = f(x)g(x, \alpha)$ suffices to illustrate the point. In the adaptive case, where the sequence of evaluation points depends on the integrand $h(x, \alpha)$, it is necessary to compute the entire integrand at all integration abscissae for each new value of the parameter α. Alternatively, each new abscissa can be stored in an array X and the corresponding value of $f(x)$ in a second array F. Then, it can be determined whether each value of x appears in the array X and if it does, $f(x)$ can be extracted from F. Otherwise, $f(x)$ is computed and then x and $f(x)$ added to the arrays X and F, respectively. This may be worthwhile if $f(x)$ is such a complicated function that its evaluation consumes much more time than a typical search in the randomly arranged array X. There

This work was supported by the Office of Naval Research, Contract Nonr 562(36).

* Present address: Department of Applied Mathematics, Weizmann Institute of Science, Rehovoth, Israel.
† Division of Applied Mathematics.

are other ways in which X may be arranged but these also are time consuming. It is clear that the programming involved in this alternative is nontrivial. On the other hand, in the Romberg case, we need to compute $f(x)$ only once at each integration abscissa, using previously computed values when they exist and computing new values as needed, without having to store the abscissae and without having to search an array each time to determine whether a given abscissa appears in it. The total number of points at which $f(x)$ is computed is equal to the maximum number of points needed to integrate $h(x, \alpha)$. A possible FORTRAN program to do this is given in Figure 1 where only the pertinent portions of the program are given. QUAD is a subroutine whose parameters are the endpoints of integration A, B and a tolerance EPS and which requires the existence of a function routine FUN(X) to compute the integrand. There is also an upper limit M to the number of points computed.

This scheme was used successfully in the computation of the following integrals arising in the solution of a problem in radiative transfer.

$$FI(\alpha) = \int_0^1 x^{-4} \exp(-R/x) E_3(\alpha x^2)\, dx$$

for $\alpha = 0(.1)3$ where $R = 10$, 12 and $E_3(x)$ is the 3rd exponential integral.

RECEIVED JUNE, 1966; REVISED JULY, 1966

```
      DIMENSION E(M), FI(N)
      COMMON ALPHA, LI, L, E
      ...
      LI = 0
      ALPHA = ALPHA1
      DO 1 I = 1, N
      L = 1
      FI(I) = QUAD(A, B, EPS)
1     ALPHA = ALPHA + ALPHA2
      ...
      END

      FUNCTION FUN(X)
      DIMENSION E(M)
      COMMON ALPHA, L1, L, E
      IF (L-L1) 1, 1, 2
2     L1 = L1 + 1
      E(L1) = F(X)
1     FUN = E(L)*G(X, ALPHA)
      L = L + 1
      RETURN
      END

      FUNCTION F(X)
      ...
      RETURN
      END

      FUNCTION G(X, ALPHA)
      ...
      RETURN
      END
```

FIG. 1. Program for the computation of $FI(\alpha) = \int_a^b f(x)g(x, \alpha)\, dx$ for α ranging from $\alpha 1$ to $\alpha 1 + (n-1)\alpha 2$ in steps of $\alpha 2$.

TABLE I. EXAMPLES OF TYPICAL ROMBERG AND ADAPTIVE SIMPSON INTEGRATION

Function $f(x)$	*Exact Value of* $\int_0^1 f(x)\, dx$	*Romberg Value for* $\epsilon = 10^{-3}, 10^{-6}$	*Number of Points*[a]	*Adaptive Simpson Value for* $\epsilon = 10^{-3}, 10^{-6}$	*Number of Points*[b]
$x^{1/2}$	0.66666667	0.66653263	65	0.66665866	55
		0.66666633	4097	0.66666655	199
$x^{3/2}$	0.40000000	0.40000854	17	0.40001016	19
		0.39999995	129	0.39999992	91
$\frac{1}{1+x}$	0.69314718	0.69314739	9	0.69314743	19
		0.69314706	33	0.69314711	55
$\frac{1}{1+x^4}$	0.86697299	0.86697292	17	0.86697326	19
		0.86697300	65	0.86697293	67
$\frac{1}{1+e^x}$	0.37988551	0.37988544	9	0.37988543	19
		0.37988546	17	0.37988543	19
$\frac{x}{e^x-1}$	0.77750463	0.77750448	9	0.77750459	19
		0.77750453	17	0.77750459	19
$\frac{2}{2+\sin 10\pi x}$	1.1547005	1.1547003	65	1.1546288	163
		1.1547004	257	1.1547002	883

[a] The minimum number of points is 9 and the number of points is always of the form 2^m+1

[b] The minimum number of points is 19 and the number of points is always of the form $7+12k$

REFERENCES

1. BAUER, F. L. Algorithm 60, Romberg integration. *Comm. ACM 4* (1961), 255; see also *Comm. ACM 5* (1962), 168, 281.
2. BAUER, F. L. La méthode d'intégration numérique de Romberg. Colloque sur l'analyse numérique, 22–24 mars 1961 à Mons, 119–129.
3. BAUER, F. L., RUTISHAUSER, H., AND STIEFEL, E. New aspects in numerical quadrature. Proc. Symposia Appl. Math. Vol. 15, 1962, 199–218.
4. BAUMANN, R., FELICIANO, M., BAUER, F. L., AND SAMELSEN, K. *Introduction to ALGOL.* Prentice-Hall, Englewood Cliffs, N. J., 1964, 73.
5. BULIRSCH, R. Bemerkungen zur Romberg-Integration. *Numer. Math. 6* (1964), 6–16.
6. BULIRSCH, R., AND STOER, J. Fehlerabschätzungen und Extrapolation mit rationalen Funktionen bei Verfahren von Richardson-Typus. *Numer. Math. 6* (1964), 413–427.
7. BULIRSCH, R., AND STOER, J. Asymptotic upper and lower bounds for results of extrapolation methods. *Numer. Math. 8* (1966), 93–104.
8. DUNKL, C. F. Romberg quadrature to prescribed accuracy. SHARE File No. 7090-1481 TYQUAD.
9. ELLIOTT, J., AND PRAGER, W. Quadrature routine to evaluate $\int_P^Q F(u)\, du$. Brown U. Computing Lab., Providence, R. I.
10. FILIPPI, S. Das Verfahren von Romberg-Stiefel-Bauer als Spezialfall des allgemeinen Prinzips von Richardson. *Mathematik-Technik-Wirtschaft 11* (1964), 49–54, 98–100.
11. GRAM, C. Definite integral by Romberg's method. ALGOL procedure. *BIT 4* (1964), 54–60; see also *BIT 4* (1964), 118–120.
12. HAVIE, T. On a modification of Romberg's algorithm. *BIT 6* (1966), 24–30.
13. KRASUN, A. M., AND PRAGER, W. Remark on Romberg quadrature. *Comm. ACM 8* (1965), 236–237.
14. KUBIK, R. N. Algorithm 257, Havie integrator. *Comm. ACM 8* (1965), 381.
15. KUNCIR, G. F. Algorithm 103, Simpson's rule integration. *Comm. ACM 5* (1962), 347.
16. MCKEEMAN, W. M. Algorithm 145, adaptive numerical

integration by Simpson's rule. *Comm. ACM 5* (1962), 604; see also *Comm. ACM 8* (1965), 171.
17. McKeeman, W. M. Certification of algorithm 145, adaptive numerical integration by Simpson's rule. *Comm. ACM 6* (1963), 167–168.
18. McKeeman, W. M., and Tesler, Larry. Algorithm 182, nonrecursive adaptive integration. *Comm. ACM 6* (1963), 315; see also *Comm. ACM 7* (1964), 244.
19. McKeeman, W. M. Algorithm 198, adaptive integration and multiple integration. *Comm. ACM 6* (1963), 443–444.
20. Meinguet, J. Methods for estimating the remainder in linear rules of approximation: application to the Romberg algorithm. Séminaire de Mathématique Appliquée et Mécanique, Université Catholique de Louvain, Rept. no. 5, 1966.
21. Prager, W. *Introduction to Basic FORTRAN Programming and Numerical Methods.* Blaisdell Publishers, New York, 1965, 123–124.
22. Romberg, W. Vereinfachte numerische Integration. *Det. Kong. Norske Videnskab. Selskab Forhandlinger, 28*, 7, 1955.
23. Rutishauser, H. Ausdehnung des Rombergscher Prinzips. *Numer. Math. 5* (1963), 48–54.
24. Stiefel, E. Altes und Neues über numerische Quadratur. *Z. Angew. Math. Mech. 41* (1961), 408–413.
25. Stiefel, E., and Rutishauser, H. Remarques concernant l'intégration numérique. *Compt. Rend. Acad. Sci. Paris 252* (1961), 1899–1900.
26. Stroud, A. H. Error estimates for Romberg quadrature. *J. SIAM Numer. Anal.*, Ser. B, *2* (1965), 480–488.
27. Thacher, Jr., H. C. Remark on algorithm 60, Romberg integration. *Comm. ACM 7* (1964), 420–421.

Automatic Error Bounds on Real Zeros of Rational Functions

Robert H. Dargel,* Frank R. Loscalzo and Thomas H. Witt
University of Wisconsin,† Madison, Wisconsin

A procedure for implementing an interval arithmetic version of the Newton-Raphson method is proposed. The procedure requires only a starting interval over which the zeros of a given rational function are to be located. The method automatically provides bounds for roundoff error.

I. Introduction

The Newton-Raphson method for determining zeros is a well-known iterative technique. Moore [3, Sec. 7-2] has proposed an interval arithmetic version which will approximate all the real zeros of a rational function of one variable in a given interval while also bounding roundoff error. A procedure is described for implementing Moore's method which has been programmed and tested on the CDC 1604 computer at the University of Wisconsin Computing Center.

II. Theoretical Background

An interval number $[a, b]$ is a set of real numbers, $\{x \mid a \leqq x \leqq b\}$. If $[a, b]$ and $[c, d]$ are two interval numbers, and if $*$ denotes one of the real arithmetic operations $+$, $-$, $\cdot$ or $\div$, then the interval arithmetic operations are defined by

$$[a, b] * [c, d] = \{x * y \mid a \leqq x \leqq b, \quad c \leqq y \leqq d\},$$

except that $[a, b] \div [c, d]$ is not defined for $0 \in [c, d]$. The following formulas, which are used in [4], satisfy the definition:

$$[a, b] + [c, d] = [a+c, b+d]$$

$$[a, b] - [c, d] = [a-d, b-c]$$

$$[a, b] \cdot [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$$

$$[a, b] \div [c, d] = [a, b] \cdot [1/d, 1/c],$$

where the division operation requires $0 \notin [c, d]$.

Rounding in interval arithmetic is executed such that the result of any interval arithmetic operation is guaranteed to contain the true answer. This is accomplished by rounding negative left endpoints and positive right endpoints, but not rounding positive left endpoints or negative right endpoints. Thus, the resultant interval is always slightly enlarged by the rounding process.

A function $F(X)$, where X is an interval, is defined to be a rational interval extension of a real rational function $f(x)$ if $F(X) \supset \{f(x) \mid x \in X\}$ and $F([x, x]) = f(x)$. Similarly, $F'(X)$ is an interval extension of $f'(x)$. Furthermore, let $m(X)$ be the midpoint of the interval X. Then Moore's method defines the function

$$N(X) = m(X) - (f(m(X))/F'(X))$$

and an interval extension of the Newton-Raphson method is provided by the recursion relation, $X_{n+1} = N(X_n) \cap X_n$. Moore [3, Lemma 7-2] has also shown that if X contains a zero of $f(x)$, then $N(X)$ also contains that zero. Conversely [3, Lemma 7-3], if $X \cap N(X)$ is empty, then X does not contain a zero of $f(x)$.

* Present address: General Dynamics, Electric Boat Div., Groton, Conn.
† Computer Sciences Department.