Gaussian Quadrature Formulae for $\int_0^1 -\ln(x)f(x)\,dx$

By Donald G. Anderson

1. Introduction. The table of Gaussian quadrature formulae for integrals of the form

$$\int_0^1 -\ln(x)f(x)\,dx$$

contained in [1] is reproduced in [2], presumably as the best available. In connection with the solution of certain nonlinear integral equations [3], I had occasion to extend this table. Presented below are a few comments on the computer program used to generate the extended table and the results obtained at that time.

2. Gaussian Quadrature Formula and Orthogonal Polynomial Generating Program. We consider quadrature formulae of the form

$$\int_a^b W(x)\,f(x)\,dx \approx \sum_{k=1}^{n} H_k^{(n)} f(x_k^{(n)}),$$

for $W(x) \geqq 0$ on $a < x < b$. Many tables of Gaussian quadrature weights $H_k^{(n)}$ and abscissae $x_k^{(n)}$, for various kernel functions $W(x)$ and intervals $a < x < b$, have appeared in the literature [4, 5, 6 and many others; see 2, 7 and references therein]. Such Gaussian formulae possess two principal advantages in the context of the numerical solution of nonlinear integral equations: first, they provide a "near-optimum" utilization of a fixed number of samples of the integrand, and second, one can treat in this fashion problems with integrably singular kernels and/or infinite intervals of integration. The difficulties inherent in generating high-order, high-precision weights and abscissae, for a given kernel and interval, inhibit one from adapting the quadrature scheme to the problem at hand unless the required tables happen to be available. Consequently, a computer program capable of generating simply and cheaply low-order ($n \sim 10$) Gaussian formulae, or composites of such formulae, is of great assistance in producing problem-oriented quadrature schemes. Such a program can be based on the algorithm summarized below—an adaption of those considered in [1, 8].

Define an inner product by

$$(f, g) = \int_a^b W(x)f(x)g(x)\,dx.$$

There exists a set of polynomials

$$p_n(x) = A_n x^n + B_n x^{n-1} + \cdots \qquad (A_n \neq 0)$$

which are mutually orthogonal with respect to this inner product, that is,

$$(p_i, p_j) = 0, \qquad \text{for } i \neq j.$$

These orthogonality conditions define the polynomials up to a multiplicative con-

Received November 9, 1964.

stant which may be chosen to normalize the p_n in any convenient manner. For a binary computer, a convenient normalization is

$$A_n = 2^{mn},$$

where the integer m can be chosen empirically, to keep (p_n, p_n) of order unity for moderate n. This normalization is easier to apply than one involving (p_n, p_n) directly, since A_n enters naturally in the recurrence relation for the p_n described below.

The quadrature sample points $x_k^{(n)}$ are the roots of $p_n(x)$; we define

$$x_0^{(n)} \equiv a \quad \text{and} \quad x_{n+1}^{(n)} \equiv b.$$

The following properties of p_n are derived in [1, 8]: The roots $x_k^{(n)}$ are real and interlaced such that

$$x_k^{(n)} < x_{k+1}^{(n+1)} < x_{k+1}^{(n)},$$

for $k = 0, 1, \cdots, n$. The quadrature weights $H_k^{(n)}$ are given by

$$H_k^{(n)} = \frac{A_n(p_{n-1}, p_{n-1})}{A_{n-1}\, p_n'(x_k^{(n)}) p_{n-1}(x_k^{(n)})}.$$

We define

$$p_0 = A_0 = 1 \quad \text{and} \quad p_1 = A_1(x - r_1).$$

The polynomials p_n satisfy a three-term recurrence relation

$$p_n(x) = \frac{A_n}{A_{n-1}} (x - r_n) p_{n-1}(x) - s_n\, p_{n-2}(x),$$

for $n = 2, 3, \cdots$. The recurrence coefficients are given by

$$r_n = u_n/t_n,$$

for $n = 1, 2, \cdots$, and

$$s_n = \frac{A_n\, A_{n-2}}{A_{n-1}^2} \frac{t_n}{t_{n-1}},$$

for $n = 2, 3, \cdots$, where

$$u_n = (xp_{n-1}, p_{n-1})$$

and

$$t_n = (p_{n-1}, p_{n-1}).$$

The particular class of quadrature formulae of interest can be defined by a subprogram which evaluates the moments of the kernel

$$M_l = (x^l, 1),$$

for $l = 0, 1, \cdots, 2n - 1$. In simple cases, the M_l can be evaluated directly as elementary functions of l; in more complicated cases, it may be necessary to approximate M_l numerically, and the program will operate with an "effective" kernel differing somewhat from the original kernel. The p_n can conveniently be represented by

$(n+1)$-vectors of their coefficients; a recurrence relation for the components of the (p_n) vector can easily be written down from that for the p_n polynomial above. If we define the ith component of the vector as $(p_n)_i$, with $(p_n)_i \equiv 0$ for $i < 0$ and $i > n$, and use the normalization condition above, we obtain

$$(p_n)_i = 2^m(p_{n-1})_{i-1} - 2^m r_n(p_{n-1})_i - s_n(p_{n-2})_i,$$

for $n = 1, 2, \cdots$; $(p_0)_0$ was defined above as unity. The inner products t_n and u_n are then given by

$$t_n = \sum_{i,j=0}^{n-1} (p_{n-1})_i(p_{n-1})_j M_{i+j},$$

$$u_n = \sum_{i,j=0}^{n-1} (p_{n-1})_i(p_{n-1})_j M_{i+j+1}.$$

It is convenient to generate the formulae successively for $n = 1, 2, \cdots$, since the root interlacing property provides bounds for the $x_k^{(n)}$ at each stage. Since p_n' is required for the evaluation of $H_k^{(n)}$, a Newton-Raphson iteration can be used to find the roots, and the $H_k^{(n)}$ follow immediately. In the critical recurrence calculations, where errors can accumulate, double precision arithmetic is desirable, even to obtain only single precision results. It is somewhat more accurate to use the recurrence relation and its derivative with respect to x to evaluate p_n and p_n', rather than using the coefficients $(p_n)_i$ and synthetic division. In the recurrence calculation, there is a tendency for a loss of significant figures in the subtraction of numbers which are approximately equal; hence, the algorithm has rather adverse roundoff error properties.

A useful byproduct of the calculation is the set of recurrence coefficients r_n, s_n, and t_n. The polynomials p_i are orthogonal under summation over the roots $x^{(k)}$ with metric $H_k^{(n)}$, since the Gaussian quadrature formula is exact for $f(x)$ a polynomial of degree $2n - 1$ or less—

$$\sum_{k=1}^{n} H_k^{(n)} p_i(x_k^{(n)}) p_j(x_k^{(n)}) = t_i \delta_{ij},$$

for $i, j = 0, 1, \cdots, n-1$, where δ_{ij} is the Kronecker delta symbol. Consequently, the p_i are often useful for generating least-squares approximations [7]. As a check on the calculation, one can compute, for each n, the maximum modulus of the difference between the orthogonality sums and their nominal values. When the program is performing properly, these test residuals contain only a few bits of roundoff error.

The basic algorithm above can be extended to facilitate the development of composite quadrature formulae made up by applying the algorithm to a set of subintervals. Perhaps the most useful extension is the inclusion of the Radau and Lobatto cases, where one or both endpoints of the interval of integration are assigned as quadrature sample points—the procedure is described in [1, 8]. Since a quadrature formula can be regarded as approximation of the kernel by a weighted sum of Dirac delta functions, one can assign quadrature weights and abscissae simultaneously. Convenient cancellations can thereby be arranged at the interfaces between sub-intervals.

As an example of the results available from such a program, consider the class of integrals of the form

$$\int_0^1 -\ln(x)f(x)\,dx.$$

For this particular kernel and interval, we obtain

$$M_l = (l+1)^{-2},$$

and the quadrature weights and abscissae of Table I. A number of other classes of integrals have been considered, but the results are not recorded here since they are not of sufficiently general interest and are easily regenerated.

There is a maximum n for which the program is useful; typically, the algorithm fails because the $x_k^{(n)}$ drift off the real axis. Even before this maximum order is reached, the results obtained will deviate from the exact quadrature weights and abscissae due to roundoff error. Nevertheless, the test residuals may still be acceptably small, and the corresponding "near-optimum" quadrature formulae acceptably accurate [9].

3. Conclusion. While a program of the class described above cannot generate high-order quadrature formulae with high precision, it can very simply and cheaply generate low-order composite formulae. Such a program is useful precisely because it is relatively inexpensive to generate quadrature schemes adapted to a particular

TABLE I

$$\int_0^1 -\ln(x)f(x)\,dx \approx \sum_{k=1}^{n} H_k^{(n)} f(x_k^{(n)})$$

n	$x_k^{(n)}$	$H_k^{(n)}$	n	$x_k^{(n)}$	$H_k^{(n)}$
2	0.11200880	0.71853931	8	0.13320243 (−1)	0.16441660
	0.60227691	0.28146068		0.79750427 (−1)	0.23752560
3	0.63890792 (−1)	0.51340455		0.19787102	0.22684198
	0.36899706	0.39198004		0.35415398	0.17575408
	0.76688030	0.94615406 (−1)		0.52945857	0.11292402
4	0.41448480 (−1)	0.38346406		0.70181452	0.57872212 (−1)
	0.24527491	0.38687532		0.84937932	0.20979074 (−1)
	0.55616545	0.19043513		0.95332645	0.36864071 (−2)
	0.84898239	0.39225487 (−1)	9	0.10869338 (−1)	0.14006846
5	0.29134472 (−1)	0.29789346		0.64983682 (−1)	0.20977224
	0.17397721	0.34977622		0.16222943	0.21142716
	0.41170251	0.23448829		0.29374996	0.17715622
	0.67731417	0.98930460 (−1)		0.44663195	0.12779920
	0.89477136	0.18911552 (−1)		0.60548172	0.78478879 (−1)
6	0.21634005 (−1)	0.23876366		0.75411017	0.39022490 (−1)
	0.12958339	0.30828657		0.87726585	0.13867290 (−1)
	0.31402045	0.24531742		0.96225056	0.24080402 (−2)
	0.53865721	0.14200875	10	0.90425944 (−2)	0.12095474
	0.75691533	0.55454622 (−1)		0.53971054 (−1)	0.18636310
	0.92266884	0.10168958 (−1)		0.13531134	0.19566066
7	0.16719355 (−1)	0.19616938		0.24705169	0.17357723
	0.10018568	0.27030264		0.38021171	0.13569597
	0.24629424	0.23968187		0.52379159	0.93647084 (−1)
	0.43346349	0.16577577		0.66577472	0.55787938 (−1)
	0.63235098	0.88943226 (−1)		0.79419019	0.27159893 (−1)
	0.81111862	0.33194304 (−1)		0.89816102	0.95151992 (−2)
	0.94084816	0.59327869 (−2)		0.96884798	0.16381586 (−2)

Note: Numbers are to be multiplied by the power of ten in parentheses.

problem. Furthermore, the programming involved is simple enough to be assigned as a laboratory exercise in a numerical analysis course. As an example, quadrature formulae adapted to a logarithmically singular kernel are given.

Harvard University
Cambridge, Massachusetts

1. H. Mineur, *Techniques de Calcul Numérique à l'Usage des Mathématiciens, Astronomes, Physiciens et Ingénieurs. Suivi de Quatre Notes Par: Mme. Henri Berthod-Zaborowski, Jean Bouzitat, et Marcel Mayot*, Béranger, Paris, 1952.
2. M. Abramowitz & I. Stegun (Eds.), *Handbook of Mathematical Functions*, National Bureau of Standards Applied Mathematics Series, v. 55, Washington, D. C., 1964; p. 920.
3. D. G. Anderson, *Numerical Experiments in Kinetic Theory*, Ph.D. Thesis, Harvard University, Cambridge, Mass., June, 1963.
4. A. N. Lowan, N. Davids & A. Levenson, "Table of the zeros of the Legendre polynomials of order 1-16 and the weight coefficients for Gauss' mechanical quadrature formula," *Bull. Amer. Math. Soc.*, v. 48, 1942, pp. 739-743. MR **4**, 90.
5. H. E. Salzer & R. Zucker, "Table of the zeros and weight factors of the first fifteen Laguerre polynomials," *Bull. Amer. Math. Soc.*, v. 55, 1949, pp. 1004-1012. MR **11**, 263.
6. H. E. Salzer, R. Zucker, & R. Capuano, "Table of the zeros and weight factors of the first twenty Hermite polynomials," *J. Res. Nat. Bur. Standards*, v. 48, 1952, pp. 111-116, MR **14**, 90.
7. P. J. Davis & P. Rabinowitz, "Advances in orthonormalizing computation," *Advances in Computers*, F. L. Alt (Ed.), Vol. 2, Academic Press, New York, 1962. MR **25** #1636.
8. F. B. Hildebrand, *Introduction to Numerical Analysis*, McGraw-Hill, New York, 1956. MR **17**, 788.
9. C. Lanczos, *Applied Analysis*, Prentice-Hall, Englewood Cliffs, N. J., 1956; p. 136. MR **18**, 823.

Quadrature Formulas Using Derivatives

By Lawrence F. Shampine

For k odd, we shall derive a new quadrature formula of the type

$$\int_{-1}^{1} f(x)\,dx \cong 2 \sum_{j=0}^{(k-1)/2} \frac{f^{(2j)}(0)}{(2j+1)!}[1 - c_j] + \sum_{l=1}^{m} a_l[f(x_l) + f(-x_l)],$$

which is exact for all polynomials of degree up to $4m + k - 2$. A similar formula holds for k even. The formulas closely resemble those of Hammer and Wicke [1]: for k odd,

$$\int_{-1}^{1} f(x)\,dx \cong 2 \sum_{j=0}^{(k-1)/2} \frac{f^{(2j)}(0)}{(2j+1)!} + \sum_{l=1}^{m} a_l[f^{(k)}(x_l) + f^{(k)}(-x_l)],$$

and a similar formula for k even. Their formulas require the use of nonclassical orthogonal polynomials. The formulas stated above are derived very simply with the use of Jacobi polynomials and would, presumably, be useful in situations similar to those envisioned by Hammer and Wicke.

$f(x)$ can be split into even and odd parts. The form of the formula is such as to integrate the odd part exactly. Let us write $f(x)$ in the form

$$f(x) = \sum_{j=0}^{\infty} \frac{f^{(2j)}(0)}{(2j)!} x^{2j},$$

Received December 3, 1964. This work was supported in part by the National Science Foundation and in part by the United States Atomic Energy Commission. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.