MATHEMATICS OF COMPUTATION, VOLUME 24, NUMBER 112, OCTOBER 1970

Control and Estimation of Computational Errors in the Evaluation of Interpolation Formulae and Quadrature Rules*

By Sven-Åke Gustafson

Abstract. Approximate rules for evaluating linear functionals are often obtained by requiring that the rule shall give exact value for a certain linear class of functions. The parameters of the rule appear hence as the solution of a system of equations. This can generally not be solved exactly but only "numerically." Sometimes large errors occur in the parameters defining the rule, but the resultant error in the computed value of the functional is small. In the present paper we shall develop efficient methods of computing a strict bound for this error in the case when the parameters of the rule are determined from a linear system of equations.

1. Introduction. In this paper we shall analyze mechanical quadrature rules and interpolation formulae which have been determined numerically by means of solving a linear system of equations. These processes can often not be carried out exactly and we want to study the errors in the computed value of the functional which thereby arise.

In Section 2 we give a general formulation of rules which can be found by using the method of undetermined coefficients and we outline a computational process which delivers a strict bound for the effect of computational errors on the value of the functional.

In the last section we treat so-called Newtonian feasible rules (defined in that section), a class of formulae which contains the Lagrangian and Hermitian rules as special cases. These rules have the pleasant property that they can be computed by a small number of multiplications and divisions. We give a general theoretical result on bounds for computational errors by the use of such rules and illustrate with examples that it is possible to solve problems in integration and summation of series in an efficient fashion by using the algorithms in [2].

2. A General Class of Linear Rules. We introduce some notations which will be used in this section. Let $[a, b]$ be a closed bounded interval and let $f; f_1, f_2, \cdots, f_n$ be $n + 1$ given functions on $[a, b]$. Further, let $L; L_1, L_2, \cdots, L_n$ be $n + 1$ given linear functionals such that $L(f)$, $L_i(f)$. $L_i(f_r)$ are all defined for $i = 1, 2, \cdots, n$, $r = 1, 2, \cdots, n$.

Put $y_r = L(f_r)$, $r = 1, 2, \cdots, n$, and let these numbers be known. Sometimes we shall call $y_1, y_2, \cdots, y_r$ moments with respect to L and the system of functions $f_1, f_2, \cdots, f_n$. This terminology is motivated by the fact that a wide class of linear

Received September 10, 1969, revised March 6, 1970.

AMS 1968 *subject classifications.* Primary 6580, 6555; Secondary 6520, 6525.

Key words and phrases. Computational errors, interpolation formulae, quadrature rules, linear rules, residuals, divided differences.

* This research was carried out at Stanford University and also supported by grants from the National Science Foundation and the Swedish Board for Technical Development.

functionals have the representation

$$L(f) = \int_a^b f(t)\, d\alpha(t)$$

and hence

$$y_r = \int_a^b f_r(t)\, d\alpha(t), \qquad r = 1, 2, \cdots, n.$$

We want to approximate L by $\mathbf{L}$, a linear combination of $L_1, L_2, \cdots, L_n$ in such a manner that $\mathbf{L}(f_r) = L(f_r)$, $r = 1, 2, \cdots, n$. Thus,

$$\mathbf{L}(f) = \sum_{i=1}^{n} m_i L_i(f), \tag{2.1}$$

where

$$\sum_{i=1}^{n} m_i L_i(f_r) = y_r, \qquad r = 1, 2, \cdots, n. \tag{2.2}$$

We shall require that the coefficient matrix of the linear system (2.2) is regular.

In order to clarify the discussion, we distinguish between two sources of error:

(a) Truncation or discretization error. This is the difference between $L(f)$ and $\mathbf{L}(f)$, an exactly computed approximation to $L(f)$.

(b) Computation error. This is the difference between $\mathbf{L}(f)$, the quantity given by a formula and the quantity which we obtain by evaluating this formula using finite precision arithmetic.

In the present paper we shall study the behaviour of computational errors (and *not* errors of the first kind).

The formulation (2.1), (2.2) applies for many familiar problems. We give some examples.

Example 2.1. A Lagrangian integration rule: Let $x_1, x_2, \cdots, x_n$ be n distinct numbers and define $L_i(f) = f(x_i)$, $i = 1, 2, \cdots, n$, and let $[a, b]$ be a closed bounded interval. Put

$$L(f) = \int_a^b f(t)\, dt.$$

Introduce further $f_r(t) = t^{r-1}$, $r = 1, 2, \cdots, n$. Then

$$y_r = \int_a^b t^{r-1}\, dt.$$

Example 2.2. An Hermitian quadrature rule: Let now n be an even number and put $n = 2k$. Select k distinct numbers $x_1, x_2, \cdots, x_k$ in the closed bounded interval $[a, b]$ and put

$$L_{2i-1}(f) = f(x_i), \qquad L_{2i}(f) = f'(x_i), \qquad i = 1, 2, \cdots, k.$$

Define L, f_r (and y_r) as in the preceding example.

Example 2.3. A Lagrangian differentiation rule: Let x be a fixed number. Define L_i and f_r as in Example 2.1, but put $L(f) = f'(x)$. Then $y_1 = 0$ and $y_r = (r - 1)x^{r-2}$, $r > 1$.

It is possible to generalize (2.1) and (2.2) further by replacing the interval $[a, b]$ by other types of sets. Therefore, the results of this section can be extended to the rules treated in [1] and [5].

We consider the general task: Let A be a regular matrix, n by n, b, x and d n-dimensional vectors. Compute

$$\gamma = d^T x, \tag{2.3}$$

when

$$Ax = b. \tag{2.4}$$

We observe that (2.1) and (2.2) is subsumed by (2.3), (2.4). We put $\gamma = \mathbf{L}(f)$, $x_i = m_i$ and $d_i = L_i(f)$, $i = 1, 2, \cdots, n$. $a_{ir} = L_i(f_r)$, $i = 1, 2, \cdots, n$, $r = 1, 2, \cdots, n$ and $b_r = y_r$, $r = 1, 2, \cdots, n$.

We prove

LEMMA 2.1. *Let A be a regular matrix, n by n, x, b, d and u n-dimensional vectors. If $\gamma = d^T x$ when $Ax = b$ then*

$$\gamma = b^T u, \tag{2.5}$$

when

$$d = A^T u. \tag{2.6}$$

Proof. From $Ax = b$ follows $x = A^{-1}b$. Hence, $\gamma = d^T x = d^T A^{-1} b = b^T (A^{-1})^T d = b^T u$ with $A^T u = d$.

In analogy to the usage in the theory of linear programming we introduce:

Definition 2.1. (2.3), (2.4) is called a primal problem; (2.5), (2.6) its dual. Hence, the dual of (2.1), (2.2) is

$$\mathbf{L}(f) = \sum_{r=1}^{n} c_r y_r, \tag{2.7}$$

$$L_i(f) = \sum_{r=1}^{n} c_r L_i(f_r), \qquad i = 1, 2, \cdots, n. \tag{2.8}$$

We establish easily that the duals of the tasks in Examples 2.1, 2.2 and 2.3 consist of the determination of certain interpolating polynomials. Next we prove

THEOREM 2.1. *Let $\bar{x}$ be an approximate solution vector of* (2.4). *Define Δx by the relation $x = \bar{x} + \Delta x$ and let $\Delta\gamma$ be the error in γ caused by replacing x with $\bar{x}$ in* (2.3). *Introduce also the residual vector ϵ given by*

$$\epsilon = b - A\bar{x}. \tag{2.9}$$

Let A be regular. Then $\Delta\gamma$ can be expressed in the following two ways:

$$\Delta\gamma = d^T \Delta x, \tag{2.10}$$

$$\epsilon = A\Delta x, \tag{2.11}$$

or

$$\Delta\gamma = \epsilon^T u, \tag{2.12}$$

$$d = A^T u. \tag{2.13}$$

Proof. Since $\bar{x}$ is assumed to be known, ϵ is computed from (2.9). Hence, the assertion is established by application of Lemma 2.1 on the dual problems (2.10), (2.11) and (2.12), (2.13).

The formulation (2.10), (2.11) can be used only if the residual vector ϵ is known with good relative accuracy. This often requires that ϵ is evaluated with arithmetic operations in a higher precision than that which was used during the solution of the main problem (2.3), (2.4). This drawback is eliminated if one uses (2.12), (2.13) for estimating a bound for the computational error.

By direct specialization of Theorem 2.1 we establish the principal result of this section:

THEOREM 2.2. *Let* (2.2) *have a regular coefficient matrix and let* $\bar{m}_i$, $i = 1, 2, \cdots, n$, *be an approximate solution of* (2.2). *Define* Δm_i *by* $m_i = \bar{m}_i + \Delta m_i$ *and let* ΔL *be the error in* $L(f)$ *caused by replacing* m_i *with* $\bar{m}_i$, $i = 1, 2, \cdots, n$. *Introduce the residuals* ϵ_r, $r = 1, 2, \cdots, n$, *given by*

$$\epsilon_r = y_r - \sum_{i=1}^{n} \bar{m}_i L_i(f_r), \qquad r = 1, 2, \cdots, n.$$

Then ΔL *can be expressed in the following two ways*

$$\Delta L = \sum_{i=1}^{n} \Delta m_i L_i(f),$$

when

$$\epsilon_r = \sum_{i=1}^{n} \Delta m_i L_i(f_r), \qquad r = 1, 2, \cdots, n,$$

or

$$\Delta L = \sum_{r=1}^{n} c_r \epsilon_r, \tag{2.14}$$

when

$$L_i(f) = \sum_{r=1}^{n} c_r L_i(f_r), \qquad i = 1, 2, \cdots, n.$$

From (2.14) we get the error bound

$$|\Delta L| \leqq \epsilon \cdot \Gamma \quad \text{where } \epsilon \geqq \max |\epsilon_r|, \qquad \Gamma = \sum_{r=1}^{n} |c_r| \tag{2.15}$$

In order to use (2.15) we need only bounds on $|\epsilon_r|$ and Γ. The latter quantity will later be referred to as the error factor. In the next section we will give a theorem which expresses Γ in terms of the higher derivatives of f if the rule defined by (2.1), (2.2) belongs to a certain class. The error analysis can be carried out in an analogous manner for the case when (2.7), (2.8) are used instead of (2.1), (2.2).

3. Newtonian Feasible Rules.

Definition 3.1. Let f_r be given by $f_r(t) = t^{r-1}$, $r = 1, 2, \cdots, n$. The solution of (2.2) is then the coefficients of a polynomial Q of degree less than n. (2.1), (2.2) are

said to define a Newtonian feasible rule if we can associate with (2.1), (2.2) n arguments (not necessarily distinct) in such a manner that Q can be expressed by means of Newton's interpolation formula with divided differences.

Example.

$$n = 6, \ \mathbf{L}(f) = m_1 f(0) + m_2 f'(0) + m_3 f''(0) + m_4 f(1) + m_5 f'(1) + m_6 f''(1).$$

This is a Newtonian feasible rule since we can introduce the six arguments: 0, 0, 0, 1, 1, 1. If f has two continuous derivatives we can express these in the form of confluent divided differences.

Counterexample.

$$n = 2, \quad \mathbf{L}(f) = m_1 f(0) + m_2 f'(1).$$

This is not a Newtonian feasible rule since we need the three arguments 0, 1, 1 to express $f(0)$ and $f'(1)$ in the form of divided differences but n is only 2. Still, (2.2) has, in this case, the unique solution $m_1 = 1$, $m_2 = 1$. We now prove the general result.

THEOREM 3.1. *Let f have n continuous derivatives on $[a, b]$ and let* (2.1), (2.2) *define a Newtonian feasible rule. Let, further, the arguments associated with the rule be $x_1, x_2, \cdots, x_n$. Define $d_1, d_2, \cdots, d_n$ by*

$$d_r = \max_{t \in I} |f^{(r-1)}(t)|/(r-1)!, \qquad r = 1, 2, \cdots, n,$$

where I is the smallest interval containing $x_1, x_2, \cdots, x_n$. If $c_1, c_2, \cdots, c_n$ is the solution of (2.4) *then*

$$\sum_{r=1}^{n} |c_r| \leqq \sum_{r=1}^{n} d_r \prod_{i=1}^{r-1} (1 + |x_i|). \tag{3.1}$$

Proof. Define Q by

$$Q(t) = \sum_{r=1}^{n} c_r t^{r-1}.$$

Since the rule is Newtonian feasible we can write Q in the form

$$Q(t) = \sum_{r=1}^{n} D_r \prod_{i=1}^{r-1} (t - x_i),$$

where D_r is a divided difference with the r arguments $x_1, x_2, \cdots, x_r$. Since f has n continuous derivatives there is a number ξ_r in I such that

$$D_r = f^{(r-1)}(\xi_r)/(r-1)!, \qquad r = 1, 2, \cdots, n.$$

Therefore, the sum of the absolute values of the coefficients of Q is less than the sum of coefficients in $\bar{Q}$ defined by

$$\bar{Q}(t) = \sum_{r=1}^{n} d_r \prod_{i=1}^{r-1} (t + |x_i|).$$

But the sum of coefficients of $\bar{Q}$ is $\bar{Q}(1)$. Hence, the assertion follows.

We observe that equality holds in (3.1), e.g., if

$$x_1 < 0, x_1 = x_2 = \cdots = x_n \quad \text{and} \quad f^{(r-1)}(x_1)/(r-1)! = d_r.$$

We conclude our analysis by discussing a few numerical examples. All of these were run on Stanford's IBM 360/67. Its Algol W compiler represents floating numbers in the form $z = x' \cdot 16^{x''}$, where x' is allotted 24 bits in single precision, 56 bits in double. Furthermore, x'' is (if possible) so selected that $1/16 \leqq |x'| \leqq 1$.

In all of our examples we work with Newtonian feasible rules. If one has to evaluate an expression in order to get input data such as abscissae and moments this is done in double precision. These data are afterwards truncated to single precision. This procedure was adopted in order to insure that the abscissae and moments were represented in full single precision, independently of the manner in which they were obtained.

The quadrature rules appearing in the examples were computed by means of the algorithms given in [2]. The error bounds were estimated according to (2.15). The residuals were computed by means of double precision arithmetic. Thus they were obtained in full relative precision.

The accumulations to form the scalar products which give the computed value of the functional were done in double precision. During this computation the fact was utilized that the product of two single precision numbers is delivered in double precision by this particular machine and compiler. It goes without saying that a more efficient use (but one more difficult to report) could have been made of the available resources. The formula

$$\Delta L \leqq \sum_{r=1}^{n} |c_r \epsilon_r|$$

derived directly from (2.14) would presumably give smaller but still strict error bounds. The computed value of the error factor Γ indicates that the total error is bounded by a rather moderate multiple of the largest residual. This could be reduced most efficiently by using double precision arithmetic during the evaluation of the weights of the pertinent quadrature rule.

Example 3.1. The integral

$$\int_0^1 \frac{1}{1+t^2}\, dt = \frac{\pi}{4}$$

was evaluated by means of Lagrangian quadrature rules with abscissae x_i, $i = 1, 2, \cdots, n$, located in the zeros of the function g defined by $g(t) = T_n(2t - 1)$, where T_n is the Chebyshev orthogonal polynomial of degree n. That is,

$$x_i = \frac{1}{2}\left[1 + \cos\left(\left(\frac{i - 0.5}{n}\right)\pi\right)\right].$$

The integrand f is given by $f(t) = 1/(1 + t^2)$ and the moments y_r by

$$y_r = \int_0^1 t^{r-1}\, dt = 1/r.$$

In this case the exact values of the weights can be computed by means of the formulae in [4, p. 127]. We report the following results.

Number of moments	Absolute value of observed maximum error in weight	Absolute value of differences between $\pi/4$ and computed result	Absolute value of largest residual	Error* factor	Estimated error bound*
3	$1.2\cdot10^{-7}$	$9.2\cdot10^{-4}$	$1.3\cdot10^{-8}$	1.55	$1.9\cdot10^{-8}$
6	$3.3\cdot10^{-6}$	$4.7\cdot10^{-6}$	$2.4\cdot10^{-7}$	3.24	$7.9\cdot10^{-7}$
9	$1.9\cdot10^{-3}$	$2.3\cdot10^{-7}$	$1.9\cdot10^{-6}$	5.52	$1.0\cdot10^{-5}$

* In this and following examples "error" refers to the error in the computed value of the functional caused by the fact that the weights of the rule are determined numerically, not exactly.

The example illustrates the fact that, although the weights are not very well determined, the bound for the contribution to the error in the computed value caused by this may be rather small. The circumstance that for 3, 6 moments the observed difference between the computed integral and $\pi/4$ is larger than the bound must be ascribed to the influence of the truncation error.

The previous example illustrates a situation where quadrature rules are well known and codes which include abscissae and weights exist. The incentive to reframe the problem ab initio is lacking.

Examples of problems for which this is not the case include integrands with numerically inconvenient weighting functions. Other examples occur in the summation of special series. For example,

Example 3.2. Evaluate $s = \sum_{r=1}^{\infty} (-1)^{r-1}\, 1/(r^2 + 1)^{\sqrt{2}-1}$.
This series belongs to the general class of series of the form

$$\sum_{r=1}^{\infty} (-1)^{r-1} a_r,$$

where a_r admits a representation

$$a_r = \int_0^1 t^{r-1}\, d\alpha(t), \qquad r = 1, 2, \cdots,$$

and the integrator α is of bounded variation over [0, 1]. α is not dependent on r. This fact can be verified by means of a table of Laplace transforms after making the substitution $t = e^{-u}$. Thus, Example 3.2 takes the form: Compute

$$\int_0^1 \frac{1}{1+t}\, d\alpha(t),$$

when

$$\int_0^1 t^{r-1}\, d\alpha(t) = \frac{1}{(r^2+1)^{\sqrt{2}-1}}.$$

Some further numerical examples are given in [3].

Royal Institute of Technology
Department of Numerical Analysis
Stockholm 70 Sweden

→ P. J. DAVIS, "A construction of nonnegative approximate quadratures," *Math. Comp.*, **v. 21**, 1967, pp. 578–582. MR **36** #5584.

2 S.-Å. GUSTAFSON, *Rapid Computation of Interpolation Formulae and Mechanical Quadrature Rules*, Technical Report, Computer Science Department, Stanford University, August 1969. (Submitted to CACM.)

3. S.-Å. GUSTAFSON, *Error Propagation by Use of Interpolation Formulae and Quadrature Rules, Which are Computed Numerically*, Technical Report, Computer Science Department, Stanford University, August 1969.

4. I. P. NATANSON, *Constructive Function Theory*, Vol III: *Interpolation and Approximation Quadratures*, GITTL, Moscow, 1949; English transl., Ungar, New York, 1965. MR **11**, 591; MR **33** #4529c.

→ M. W. WILSON, "A general algorithm for nonnegative quadrature formulas," *Math. Comp.*, **v. 23**, 1969, pp. 253–258. MR **39** #3705.