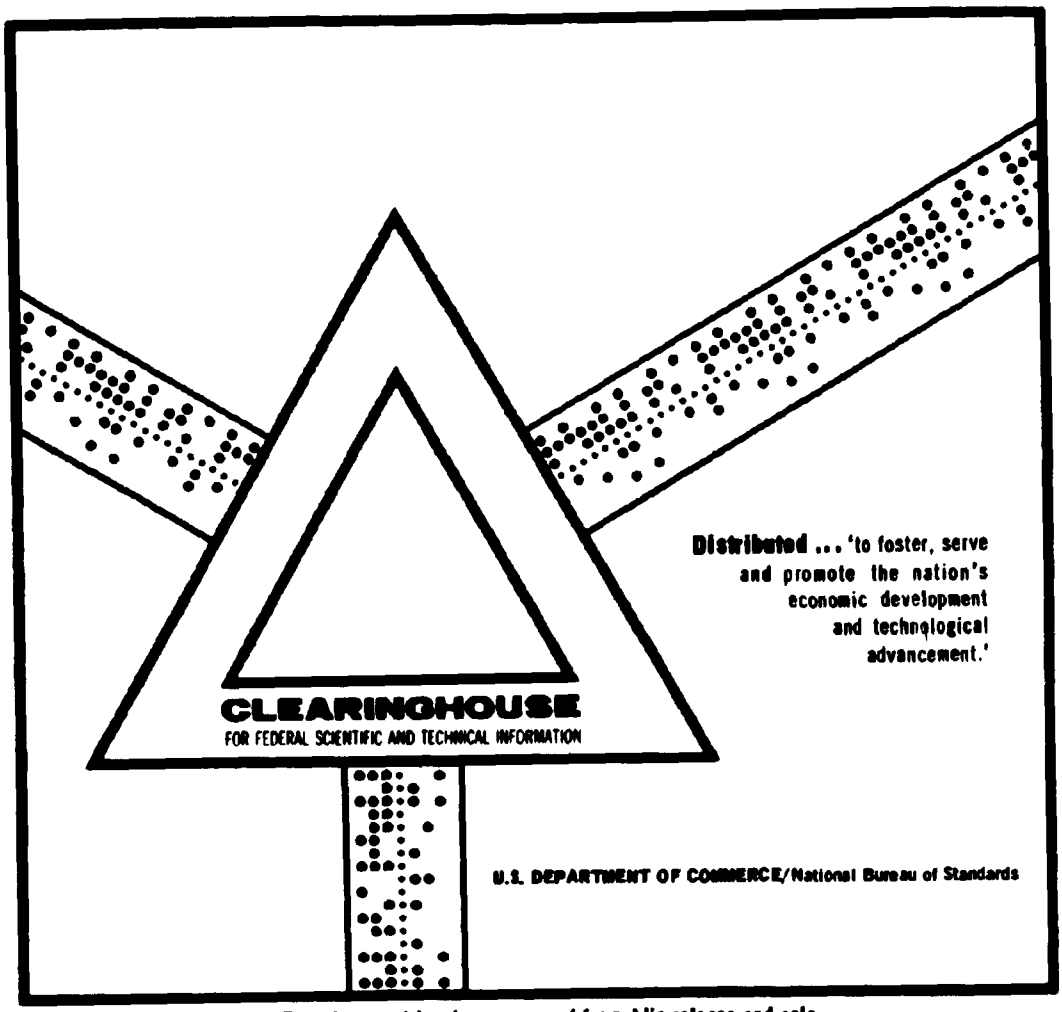
ERROR PROPAGATION BY USE OF INTERPOLATION FORMULAE AND QUADRATURE RULES ARE COMPUTED NUMERICALLY

Sven-Ake Gustafson

Stanford University
Stanford, California

February 1970

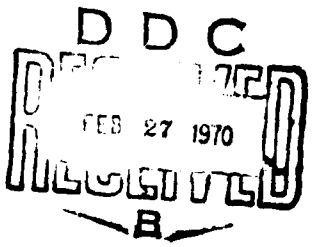


ERROR PROPAGATION BY USE OF INTERPOLATION FORMULAE AND QUADRATURE RULES WHICH ARE COMPUTED NUMERICALLY

BY

SVEN-AKE GUSTAFSON

STAN-CS-70-153
FEBRUARY 1970



COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Error propagation by use of interpolation formulae and quadrature rules which are computed numerically.

by

Sven-Åke Gustafson

This research was supported financially by grants from the National Science Foundation and the Swedish Government.

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. This process can often not be carried out exactly and we want to study the errors in the computed value of the functional which hereby arise.

In section 2 we give a general formulation of rules which can be found by using the method of undetermined coefficients and outline a computational process which delivers a strict error bound in an economical manner.

In the last section we treat so-called Newtonian feasible rules (See [7]), 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 error bounds for 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 [7].

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 $f; f_1, f_2, \ldots, f_n$ be $n + 1$ given functions on $[a,b]$. Further, let $L; L_1, L_2, \ldots, L_n$ be $n + 1$ given linear functionals such that $L(f)$, $L_i(f)$, $L_i(f_r)$ are all defined for $i = 1, 2, \ldots, n$, $r = 1, 2, \ldots, n$.

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

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

and hence

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

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

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

where

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

We must require that the linear system (2.2) has a solution. The formulation (2.1), (2.2) applies for many familiar problems. We give some examples.

Example 2:1 A Lagrangian integration rule: Let $x_1, x_2, \ldots, x_n$ be n distinct numbers and define $L_i(f) = f(x_i)$, $i = 1, 2, \ldots, n$ and put.

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

Introduce further $f_r(t) = t^{r-1}$, $r = 1, 2, \ldots, n$. 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 $x_1, x_2, \ldots, x_k$ and put

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

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

Example 2:3 A Lagrangian derivation 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 $y_1 = 1$ and $y_r = (r-1)x^{r-2}$, $r > 1$.

It is possible to obtain general rules by appropriate selection of the function system $f_1, f_2, \ldots, f_n$. An obvious generalization is to replace the interval $[a,b]$ by other types of sets. Therefore one can extend (2.1), (2.2) to the rules treated, e.g. in [2] and [9].

If the coefficient matrix of (2.2) is regular we can replace (2.1), (2.2) with an algebraically equivalent problem. Let namely A be a regular matrix, n by n and b, c, x and u n-dimensional column vectors. Then (2.1), (2.2) is a special case of the task: Evaluate $\gamma = c^T x$ when $Ax = b$. However, we find immediately that we can also write $\gamma = b^T u$ when $A^T u = c$. Using this observation we can replace (2.1), (2.2) with the alternative formulation

$$\underline{L}(f) = \sum_{r=1}^{n} c_r y_r \tag{2.3}$$

when

$$\sum_{r=1}^{n} c_r L_i(f_r) = L_i(f) \ , \ i = 1, 2, \ldots, n \tag{2.4}$$

In analogy to the usage in the theory of linear programming we shall call (2.1), (2.2) a primal problem, (2.3) and (2.4) its dual.

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.

We now want to derive general error bounds by using the dual problems introduced above. Assume that (2.2) has been solved numerically yielding the approximation $\bar{m}_i$ for m_i, $i = 1, 2, \ldots, n$. Define Δm_i by $\bar{m}_i = m_i + \Delta m_i$ and let ΔL be the error in $\underline{L}(f)$ caused by using $\bar{m}_i$

instead of m_i , $i = 1, 2, \ldots, n$. Introduce also the residuals ε_r , $r = 1, 2, \ldots, n$ given by

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

Hence

$$\Delta L = \sum_{i=1}^{n} \Delta m_i L_i(f) \tag{2.5}$$

when

$$\sum_{i=1}^{n} \Delta m_i L_i(f_r) = \varepsilon_r \ , \ r = 1, 2, \ldots, n \ . \tag{2.6}$$

The dual of this problem reads:

$$\Delta L = \sum_{r=1}^{n} c_r \varepsilon_r \tag{2.7}$$

when

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

The formulation (2.5), (2.6) can be used only if the residuals are known with good relative accuracy. This often requires that they are evaluated by means of arithmetic operations in a higher precision than that which was used during the solution of (2.2). This drawback can be eliminated if one uses (2.7), (2.8) instead. From (2.7) we get the error bound

$$(2.9) \qquad |\Delta L| \leq \varepsilon \cdot \Gamma \quad \text{where} \quad \varepsilon \geq \max|\varepsilon_r| \ , \ \Gamma = \sum_{r=1}^{n} |c_r|$$

In order to use (2.9) we need only bounds on $|\varepsilon_r|$ 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.

Generally the error factor is most easily found by solving (2.8) which can be done without too much effort. We observe namely that the coefficient matrix of (2.8) is the transpose of that of (2.2). Therefore if we solve the latter system by means of Gaussian elimination with pivoting we obtain a partition of its coefficient matrix which can be utilized for the subsequent solution of (2.8). Hence this system can be solved by means of about n^2 operations. (We use the word "operation" for a multiplication or a division.) Since the residuals can be evaluated by means of n^2 operations the total number of operations to obtain a rule of the type (2.1), (2.2) and an error bound can be written

$$(1 + 6/n)(n^3/3 + O(n^2))$$

The error analysis can be carried out in an analogous manner for the case when (2.3), (2.4) are used instead of (2.1), (2.2).

3. Newtonian feasible rules

In this section we shall treat the case when f_r is defined by

$$f_r(t) = t^{r-1} \quad , \quad r = 1, 2, \ldots, n$$

Then the solution of (2.2) is the coefficients of a polynomial Q of degree less than n. We shall also require that 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 formula with divided differences with respect to these arguments. A rule meeting these conditions will be termed a Newtonian feasible rule (this definition is equivalent with that in [7]).

Example

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

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.

Counter-example

$$n = 2 \qquad \underline{L}(f) = m_1 f(0) + m_2 f'(1)$$

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 theorem:

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 $x_1, x_2, \ldots, x_n$. Define $d_1, d_2, \ldots, d_n$ by

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

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

$$\sum_{r=1}^{n} |c_r| \leq \sum_{r=1}^{n} d_r \prod_{j=1}^{r-1} (1 + |x_j|) \ . \tag{3.1}$$

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 under the form

$$Q(t) = \sum_{r=1}^{n} D_r \prod_{j=1}^{r-1} (t - x_j)$$

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

$$D_r = f^{(r-1)}(\xi_r)/(r-1)! \quad , \; r = 1, 2, \ldots, n$$

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

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

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

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 $1/16 \leq |x'| \leq 1$.

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 [7]. The error bounds were estimated according to (2.9).

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 (but more difficult to report) use could have been done of the available resources. The formula

$$\Delta L \leq \sum_{r=1}^{n} |c_r \varepsilon_r|$$

derived directly from (2.7) would presumably give smaller but still strict error bounds. The computed value of the error factor Γ indicate that the total error is bounded by a rather moderate multiple of the largest residual. This could be brought down 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, ..., n , located in the zeros of the function g defined by $g(t) = T_n(2t + 1)$ where T_n is the Čebyšev 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 $f(t) = \frac{1}{1 + t^2}$ and hence the moments y_r

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

In this case the exact values of the weights can be computed by means of the formulae in [8], page 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	$1.2 \cdot 10^{-7}$	$9.2 \cdot 10^{-4}$	$1.3 \cdot 10^{-8}$	1.55	$1.9 \cdot 10^{-8}$
6	$3.3 \cdot 10^{-6}$	$4.7 \cdot 10^{-6}$	$2.4 \cdot 10^{-7}$	3.24	$7.9 \cdot 10^{-7}$
9	$1.9 \cdot 10^{-3}$	$2.3 \cdot 10^{-7}$	$1.9 \cdot 10^{-6}$	5.53	$1.0 \cdot 10^{-5}$

*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 three 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.

Example 3:2 Compute $\int_0^1 \frac{e^{\frac{1}{4+\sin t}} \ln(1/t)}{1+t}\, dt$.

This example illustrates how a suitable choice of a weight function can result in accurate quadrature rules. These latter are computed with the algorithms described in [7]. In this case we take the integrand f defined by

$$f(t) = e^{\frac{1}{4+\sin t}}$$

Hence the moments $y_1, y_2, \ldots, y_n$ are

$$y_r = \int_0^1 \frac{t^{r-1}\ln(1/t)}{1-t}$$

They are obtained by the recurrence relation

$$y_1 = \pi^2/12 \ , \ y_r + y_{r+1} = 1/r^2 \ , \ r = 1, 2, \ldots .$$

Lagrangian rules with abscissae as in example 3:1 were used. We give the results

Number of moments	Computed value of integral	Absolute value of largest residual	Error factor	Error bound
2	1.04370	$6.2\cdot10^{-8}$	1.34	$8.3\cdot10^{-8}$
3	1.04362	$5.3\cdot10^{-8}$	1.39	$7.4\cdot10^{-8}$
4	1.04362	$6.2\cdot10^{-8}$	1.39	$8.7\cdot10^{-8}$

We observe that the error which can be caused by inaccurate weights is neglible in comparison to the working precision.

Example 3:3 Evaluate $s = \sum_{r=1}^{\infty} (-1)^{r-1} \frac{1}{(r^2+1)^{\sqrt{2}-1}}$

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

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

and the integrator α is of bounded variation over $[0,1]$. α is not dependent on r. This fact can sometimes be verified by means of a table of Laplace transforms after making the substitution $t = e^{-u}$. Thus example 3:3 takes the form

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

when

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

We use again Lagrangian rules with abscissae allocated as in example 1. We report the results

Number of moments	Computed sum	Absolute value of largest residual	Error factor	Error bound
2	0.4785 503	$1.2\cdot10^{-8}$	1.41	$1.7.10^{-8}$
4	0.4819 880	$3.2\cdot10^{-8}$	2.83	$9.1\cdot10^{-8}$
6	0.4821 010	$5.7\cdot10^{-8}$	4.24	$2.4.10^{-7}$
8	0.4821 032	$6.0\cdot10^{-8}$	5.65	$3.4\cdot10^{-7}$
10	0.4821 032	$1.3\cdot10^{-7}$	6.88	$9.3.10^{-7}$

Example 3:4 Evaluate $s = \sum_{r=1}^{\infty} (-1)^{r-1} e^{-\sqrt{r}}$

This series belongs to the subset of the general class discussed in the preceding example where the numbers a_r introduced there are the moments of an nondecreasing integration α. That is we can write

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

when

$$\int_0^1 t^{r-1}\, d\alpha(t) = e^{\sqrt{r}}\ ,\ r = 1, 2, \ldots$$

and in addition α' .

As shown in [1] the integral is bounded between the values which result if certain quadrature rules of Gaussian type are applied to the integral for s . These rules can be computed by algorithms given in [5], [3], [4], and [6]. One must, however, solve a non-linear system of equations which is unique for each series. This is avoided in the following way: Since the derivatives of the function f defined by

$$f(t) = (1+t)^{-1}$$

do not change sign we can by means of Hermite-interpolation construct two polynomials P_1 and P_2 such that

$$P_1(t) \leq f(t) \leq P_2(t)$$

Let n be the number of moments needed in the quadrature rules and let x_i and t_i be distinct points in (0,1). The conditions defining P_1 and P_2 must be of the form:

$$n = 2k+1$$

$$\begin{aligned} P_1(1) &= f(1) & P_2(0) &= f(0) \\ P_1(t_i) &= f(t_i) & P_2(x_i) &= f(x_i) \qquad i = 1, 2, \ldots, k \\ P_1'(t_i) &= f'(t_i) & P_2'(x_i) &= f'(x_i) \end{aligned}$$

$$n = 2k$$

$$P_1(t_i) = f(t_i) \qquad P_2(x_i) = f(x_i)$$
$$P_1'(t_i) = f'(t_i) \qquad P_2'(x_i) = f'(x_i) \qquad i = 1, 2, \ldots, k$$
$$P_2(0) = f(0)$$
$$P_2(1) = f(1)$$

Thus we only have to solve linear systems of equations (which can be done with n^2 operations). Since P_1 and P_2 do not depend on the moment sequence the same P_1 and P_2 applies to all series of the class discussed here. In example 3:3 we computed upper and lower bounds by allocating the interior touching points in the zeros of Čebyšev $T_k(2x+1)$ where T_k is a Čebyšev polynomial with degree equal to the number of such points. We give the results

Number of moments	Difference between bounds	Estimated error in upper bound	Estimated error in lower bound	Difference between computed bounds Gaussian rules*
3	$5.7 \cdot 10^{-3}$	$1.3 \cdot 10^{-7}$	$1.1 \cdot 10^{-7}$	$2.5 \cdot 10^{-3}$
6	$4.9 \cdot 10^{-5}$	$8.0 \cdot 10^{-7}$	$3.2 \cdot 10^{-7}$	$9.9 \cdot 10^{-6}$
9	$3.0 \cdot 10^{-7}$	$1.7 \cdot 10^{-6}$	$6.9 \cdot 10^{-7}$	$4.0 \cdot 10^{-8}$

*For this computation double precision was used throughout.

References

[1] Dahlquist, G., Gustafson, S.-Å., and Siklósi, K., Convergence acceleration from the point of view of linear programming, BIT (5), 1965, p. 1-16.

[2] Davis, P. J., A construction of non-negative approximate quadratures, Math. of Comp., Vol 21 (1967), p. 578-582.

[3] Gautschi, W., Construction of Gauss-Christoffel quadrature formulas, Math. of Comp., Vol. 22 (1968), p. 251-270.

[4] Gautschi, W., Alg 331, Gaussian quadrature formulas, CACM, Vol. 11 (1968), p. 432-436.

[5] Golub, G. H., Welsch, J. H., Calculation of Gauss quadrature rules, Math. of Comp., Vol. 23 (1969), p. 221-230.

[6] Gustafson, S.-Å., Algorithm D1 CAL GAB, write-up, Comp. Center, University of California, Berkeley, Feb. 1969.

[7] Gustafson, S.-Å., Rapid computation of interpolation formulae and mechanical quadrature rules, Technical Rep., Computer Science Department, Stanford University, August 1969. (Submitted to CACM)

[8] Natanson, I. P., Constructive function theory, Vol. III: Interpolation and quadratures, F. Ungar Publishing Co., New York, 1965.

[9] Wilson, M. W., A general algorithm for non-negative quadrature formulas, Math. of Comp., Vol. 23 (1969), p. 253-258.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)
Computer Science Department
Stanford University
Stanford, California 94305

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
--

3 REPORT TITLE
ERROR PROPAGATION BY USE OF INTERPOLATION FORMULAE AND QUADRATURE RULES WHICH ARE COMPUTED NUMERICALLY

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)
Manuscript for Publication (Technical Report)

5 AUTHOR(S) (First name, middle initial, last name)
Sven-Åke Gustafson

6 REPORT DATE
February 1970

7a. TOTAL NO. OF PAGES
17

7b. NO. OF REFS
9

8a. CONTRACT OR GRANT NO
N00014-67-A-0012-0029

b. PROJECT NO.
NR 004-211

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
STAN-CS-70-153

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
none

10. DISTRIBUTION STATEMENT
Releasable without limitations on dissemination.

11. SUPPLEMENTARY NOTES
---

12. SPONSORING MILITARY ACTIVITY
Office of Naval Research

13 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.

DD FORM 1 NOV 68 1473 (PAGE 1)
S/N 0101-807-6801

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
method with undetermined coefficients						
linear functional						
error bound						
Newton's interpolation formula with divided differences						