BULLETIN OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 80, Number 6, November 1974

# ADAPTIVE QUADRATURE: CONVERGENCE OF PARALLEL AND SEQUENTIAL ALGORITHMS

BY JOHN R. RICE[1]

Communicated by A. S. Householder, May 28, 1974

Adaptive quadrature algorithms dyamically choose the weights and abscissae in the formula

$$\int_0^1 f(x)\,dx \approx Q_N f = \sum_{i=1}^{N} w_i f(x_i)$$

to adapt their estimates to the particular nature of the integrand $f(x)$. Within the past five years experimental evidence has appeared to suggest that adaptive quadrature algorithms are significantly superior to traditional quadrature formulas because they have a much wider domain of efficient applicability with little sacrifice in computational effort.

A metalgorithm is an abstraction representing a large class of algorithms and is used to discuss and analyze the properties of adaptive quadrature algorithms. A novel feature of these algorithms is the important role that data structures (for the interval collection) play in their behavior. A study of reasonable selections of components for the metalgorithm shows that there are from 1 to 10 *million* potentially interesting, significantly distinct adaptive quadrature algorithms. This situation illustrates the extreme difficulty of the common problem of selecting the "best" algorithm for a particular computation.

Our purpose is to announce convergence results for a variety of adaptive quadrature algorithms[2] (including all 10 million mentioned above). Roughly

*AMS (MOS) subject classifications* (1970). Primary 65D30.

[1]This work partially supported by NSF grant GP-32940X.

[2]This announcement is a summary of results and analysis contained in the following:

*A metalgorithm for adaptive quadrature,* CSD-TR 89, Computer Science Department, Purdue University, March, 1973, 43pp. J. Assoc. Comput. Mach. (to appear).

*Parallel algorithms for adaptive quadrature: Convergence,* CSD-TR 104, Computer Science Department, Purdue University, September, 1973, 18pp. Proc. IFIP 74 (to appear).

*Parallel algorithms for adaptive quadrature* II; *Metalgorithm correctness,* CSD-TR 107, November, 1973, 28pp. (to appear).

*Parallel algorithms for adaptive quadrature* III; *Program correctness,* CSD-TR 112, March, 1974, 52pp. (to appear).

speaking, these theorems state that adaptive quadrature algorithms are as efficient and effective for badly behaved integrands (finite number of singularities) as comparable traditional quadrature formulas are for well-behaved integrands. This result explains the experimentally observed superiority of adaptive algorithms.

The key to the analysis is the fact that an algorithm generates a tree of intervals as it adaptively subdivides the elements of the interval collection (subdivision is usually into two equal halves). This subdivision naturally decreases the error in the quadrature estimates, and we abstract the process as follows: We are given:

A Numbers $\epsilon > 0$ and $\gamma, \beta < 1$.

B An empty set $M'$ and a set $M$ of intervals $I$ with associated numbers $\eta(I)$. $M$ contains a distinguished interval $I^*$.

C A process $P: I \longrightarrow (IL, IR)$ which divides $I$ into left and right subintervals $IL$ and $IR$ such that:

(i) If $I = I^*$ then $\eta(IL) = \eta(IR) = \beta^*\eta(I)$ and $I^* \leftarrow IL$ or $I^* \leftarrow IR$.

(ii) If $I \neq I^*$ then $\eta(IL) = \eta(IR) = \gamma^*\eta(I)$.

We now define a *partition algorithm*:

*For* $I \in M$ *do*

$P: I \longrightarrow (IL, IR)$
*If* $(\eta(IL) < \epsilon)$ *then* $IL \in M'$ *else* $IL \in M$
*If* $(\eta(IR) < \epsilon)$ *then* $IR \in M'$ *else* $IR \in M$

This algorithm terminates when $M$ is empty and the following theorem estimates the resulting size of $M'$.

THEOREM 1. *Consider the partition algorithm with* $\beta$, $M$ *and* $\eta(I)$ *for* $I \in M$ *fixed initially. Let* $F(\gamma, \delta)$ *be the size of* $M'$ *when the algorithm terminates and we have*

$$F(\gamma, \epsilon) = O(\epsilon^{1/\log_2 \gamma}).$$

It is believed that this result is the key to the analysis of adaptive algorithms of other types (e.g., curve fitting, solutions of integral or differential equations) as well as for quadrature.

To apply this result, we introduce two assumptions of a traditional mathematical nature.

ASSUMPTION 1 (*Integrand*). *Let* $f(x)$ *have singularities* $S = \{s_i | i = 1, 2, \cdots, R < \infty\}$ *and set* $w(x) = \Pi_{i=1}^{R}(x - s_i)$.

(i) *If* $x_0 \notin S$ *then* $f^{(p)}(x)$ *is continuous in a neighborhood of* $x_0$.

(ii) *There are constants* $K$, $\alpha$ *and* $p \geq 2$ *so that*

$$|f^{(p)}(x)| \leq K|w(x)|^{\alpha-p}.$$

We assume for simplicity that intervals are subdivided into two parts and let $E(x, k)$ denote the algorithm's bound on the quadrature error for the interval $[x, x + 2^{-k}]$.

ASSUMPTION 2 (*Quadrature formula*). *With the constants* $p$, $K$ *and* $\alpha$ *of Assumption* 1 *we have*

(i) *If* $[x, x + 2^{-k}]$ *contains no singularity, then*

$$E(x, k) \leq K \max_{[x,x+2^{-k}]} |f^{(p)}(x)|2^{-pk};$$

*otherwise* $E(x, k) \leq K2^{-\alpha k}$.

(ii) *The quadrature formula for one interval requires* $q$ *or fewer evaluations of* $f(x)$.

We now must define the algorithms to be considered, and for simplicity we assume that the data structure for the interval collection consists of two boxes. A value $\epsilon > 0$ is specified and an interval is in the active box if $E(x, k) \geq \epsilon$, otherwise it is in the discard box. Intervals are chosen for processing by any means whatsoever from the active box. One may interpret more realistic data structures (e.g., stacks, queues, ordered lists) in terms of this simplistic one and, hence, apply the convergence result to realistic algorithms. We say we have a 2-*box adaptive quadrature algorithm* if its processor satisfies Assumption 2 and it uses the two-box data structure. Recall that $N$ is the number of integrand evaluations.

THEOREM 2. *Let* $f(x)$ *satisfy Assumption* 1 *with* $\alpha > -1$. *For a* 2-*box algorithm we have, as* $N \to \infty$,

$$\left| \int_0^1 f(x)\, dx - Q_N f \right| = O(1/N^p).$$

Note that this is the same conclusion that we obtain for traditional quadrature rules for $f(x) \in C^p[0, 1]$ and yet it includes $f(x) = x^{-3/4}$.

We have considered the following questions for parallel computation:

1. Does Theorem 2 extend to parallel algorithms?
2. Can mathematical convergence be replaced by algorithmic convergence?
3. Can one prove an algorithmic convergence result for a specific computer program?

4. What is the nature of the speedup in the computation that one obtains with parallel computation?

The answer to the first question is yes, the metalgorithm and the proofs can be modified and extended so as to apply to parallel computations. We explain the nonstandard distinction between mathematical and algorithmic convergence. Ordinary mathematical convergence states that an algorithm eventually produces accurate quadrature estimates.

Note that every sequence of numbers printed by a computer is the sequence of quadrature estimates for Simpson's rule applied to some $f(x) \in C^4[0, 1]$ and, thus, there is no way to decide when to accept one of the estimates as accurate. Algorithmic convergence occurs when an algorithm is given an arbitrary $\epsilon > 0$ and it terminates with an estimate with error less than $\epsilon$. In computation it is algorithmic and not mathematical convergence that is of interest, but it cannot be achieved without an additional assumption on $f(x)$. We introduce the characteristic length $\lambda(f)$ of a function $f(x)$ for a particular algorithm and $\lambda(f)$ is such that: *when an interval's length is less than* $\lambda(f)$, *then* $E(x, k)$ *is a true bound on the quadrature error.* It may be nontrivial to discover $\lambda(f)$ for a particular $f(x)$ and algorithm or to show that $\lambda(f) > 0$ exists for any reasonable class of functions. It has been shown that the required characteristic lengths of two well-known algorithms, SQUANK [2] and CADRE [1] are .5 and 1, respectively.

We have created a program PAFAQ for a parallel computer for which the characteristic length is input and readily determined in most cases; $\lambda(f)$ is the minimum separation of inflection points and singularities. We have proved that this program is correct and that it has algorithmic convergence of the order specified in Theorem 2. The convergence proof is in three levels. The first and most abstract is the extension of the sequential result to parallel computation. The second level involves a much more specific metalgorithm which includes a set of 39 specific attributes. It is at this point that the change from mathematical to algorithmic convergence takes place, and it is shown that every algorithm represented by this metalgorithm has a rate of convergence (in the algorithmic sense) as specified in Theorem 2. The third level is the only one to involve the computer program, and it is shown both that the program is correct and that it is a specific algorithm represented by the second level metalgorithm. The convergence result then follows immediately. The program is written for a hypothetical (but very reasonable) parallel computer with a number of asynchronously operating general purpose processors.

The ideal speedup for a computer with $M$ processors is $1/M$, i.e. if it takes 1 unit of time in a sequential computation, then it takes $O(1/M)$ units of time in parallel. We have not been able to achieve anything near this ideal, and, in fact, have only been able to show that there is constant factor of speedup possible. We believe that much better results can be established, probably $O(\log M/M)$. The slow down in the computation is due to the time required to obtain and return intervals from and to the interval collection. This must be done *very* carefully, otherwise two processors obtain the same interval or one interval gets lost and the computation is ruined.

## REFERENCES

1. C. de Boor, CADRE: *An algorithm for numerical quadrature,* Mathematical Software (J. R. Rice, ed.), Academic Press, 1971, pp. 417–449.

2. J. N. Lyness, SQUANK (*Simpson quadrature used adaptively–noise killed*), Algorithm 379, Comm. ACM **13** (1970), 260–263.

DEPARTMENT OF MATHEMATICS, PURDUE UNIVERSITY, WEST LAFAYETTE, INDIANA 47907