SOFTWARE FOR NUMERICAL MATHEMATICS

Proceedings of the Loughborough University of Technology Conference of the Institute of Mathematics and Its Applications held in April 1973

Edited by

D. J. EVANS

SOFTWARE FOR NUMERICAL MATHEMATICS

Proceedings of the Loughborough University of Technology Conference of the Institute of Mathematics and Its Applications held in April 1973

Edited by

D. J. EVANS

Department of Mathematics
Loughborough University of Technology
Loughborough, Leicestershire, England

ACADEMIC PRESS LONDON AND NEW YORK

A Subsidiary of Harcourt Brace Jovanovich, publishers

ACADEMIC PRESS INC. (LONDON) LTD.
24/28 Oval Road,
London NW1

United States Edition published by
ACADEMIC PRESS INC.
111 Fifth Avenue
New York, New York 10003

Library of Congress Catalog Card Number: 73-18998
ISBN: 0-12-243750-0

Printed in Great Britain by
ROYSTAN PRINTERS LIMITED
Spencer Court, 7 Chalcot Road
London NW1

Contributors

J. M. BOYLE; *Applied Mathematics Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, U.S.A.*

C. W. CLENSHAW; *Department of Mathematics, Cartmel College, Bailrigg, Lancaster LA1 4YR, Lancashire, England.*

W. J. CODY; *Applied Mathematics Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, U.S.A.*

M. G. COX; *Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex, TW11 0LW, England.*

L. C. W. DIXON; *Numerical Optimisation Centre, The Hatfield Polytechnic, 19 St. Albans Road, Hatfield, Hertfordshire, England.*

VALERIE A. DIXON; *Oxford University Computing Laboratory, 19 Parks Road, Oxford, OX1 3PL and Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex, TW11 0LW, England.*

B. EINARSSON; *Försvarets Forskningsanstalt, The Research Institute of National Defence, Box 98, S-14700 Tumba, Sweden.*

D. J. EVANS; *Department of Mathematics, Loughborough University of Technology, Loughborough, Leicestershire, LE11 3TU, England.*

R. FLETCHER; *Theoretical Physics Division, U.K.A.E.A. Research Group, Atomic Energy Research Establishment, Harwell, Didcot, Berkshire, OX11 ORA, England.*

B. FORD; *Numerical Algorithms Group, Oxford University Computing Laboratory, 19 Parks Road, Oxford, OX1 3PL, England.*

A. C. GENZ; *Mathematical Institute, Cornwallis Building, The University of Kent at Canterbury, Kent, England.*

S. J. HAGUE; *Numerical Algorithms Group, Oxford University Computing Laboratory, 19 Parks Road, Oxford, OX1 3PL, England.*

J. G. HAYES; *Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex, TW11 0LW, England.*

M. D. HEBDEN; *Theoretical Physics Division, U.K.A.E.A. Research Group, Atomic Energy Research Establishment, Harwell, Didcot, Berkshire, OX11 0RA, England.*

D. HUTCHINSON; *Centre for Computer Studies, University of Leeds, Leeds, LS2 9JT, England.*

P. JESTY; *Department of Management Studies, Leeds Polytechnic, Leeds, England.*

SHIRLEY A. LILL; *Computer Laboratory, The University of Liverpool, Brownlow Hill and Crown Street, Liverpool, L69 3BX, England.*

G. F. MILLER; *Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex, TW11 OLW, England.*

G. MITRA; *Department of Statistics and Operational Research, Brunel University, Kingston Lane, Uxbridge, UB8 3PH, Middlesex, England.*

M. J. D. POWELL; *Theoretical Physics Division, U.K.A.E.A. Research Group, Atomic Energy Research Establishment, Harwell, Didcot, Berkshire, OX11 0RA, England.*

J. A. PRENTICE; *Computing Centre, Loughborough University of Technology, Loughborough, Leicestershire, LE11 3TU, England.*

J. K. REID; *Theoretical Physics Division, U.K.A.E.A. Research Group, Atomic Energy Research Establishment, Harwell, Didcot, Berkshire, OX11 0RA, England.*

J. L. SCHONFELDER; *Computer Centre, The University of Birmingham, P.O. Box 363, Birmingham, B15 2TT, England.*

B. T. SMITH; *Applied Mathematics Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, U.S.A.*

A. SYKES; *Theory Division, Culham Laboratory, U.K.A.E.A. Research Group, Abingdon, Berkshire, OX14 3DB, England.*

D. B. TAYLOR; *Edinburgh Regional Computing Centre, Edinburgh University, Edinburgh, EH8 9YL, Scotland.*

C. LL. THOMAS; *Theory Division, Culham Laboratory, U.K.A.E.A. Research Group, Abingdon, Berkshire, OX14 3DB, England.*

J. F. TRAUB; *Department of Computer Science, Carnegie-Mellon University, Schenley Park, Pittsburgh, Pennsylvania 15213, U.S.A.*

JOAN E. WALSH; *Department of Mathematics, University of Manchester, Manchester, M13 9PL, England.*

J. H. WILKINSON; *Division of Numerical Analysis and Computing, National Physical Laboratory, Teddington, Middlesex, TW11 0LW, England.*

Preface

The contents of this book are based on lectures and discussions given at the 'Software for Numerical Mathematics' conference held at Loughborough University of Technology in April 1973 under the sponsorship of the Institute of Mathematics and Its Applications. The conference was attended by some 220 participants drawn from fifteen different countries representing both academic and industrial interests.

The aims of the conference were to provide a forum for the exchange of ideas and information on the analysis, development, construction, evaluation, communication and usage of numerical algorithms—a rapidly expanding discipline intended to support the many areas of computer applications in mathematics, science and engineering. It was thought that the most effective way of emphasising attention to this important area was by bringing together in a meeting such as this, people of relevant experience currently making everyday contributions to the field.

In the conference organisation, I was ably assisted by a committee which consisted of J. R. A. Cooper, B. Ford, A. R. Gourlay and M. J. D. Powell who discharged their duties by arranging a conference programme divided into sessions in which eminent speakers were invited to give survey lectures to cover the basic issues of the topic, whilst a smaller number of research papers of a specialist nature was selected from people wishing to participate in the conference. After each presentation the floor was open for informal discussions in order to achieve as much cross fertilisation of ideas as possible.

These proceedings could not have been published so quickly without the co-operation of the lecturers who presented their papers lucidly and made them available for publication on time. Essentially, the content of each paper is the responsibility of the author concerned although I have made slight changes where necessary to aid clarity and presentation. The discussions, with some editing, have also been included to complete the proceedings.

I should like to conclude by acknowledging the support given by the Institute of Mathematics and Its Applications for the skilful arrangement of the many financial and domestic details in the administration and organisation of the conference, the Conference Committee, the Session Chairmen and to Dr. J. Wilkinson, F.R.S. and Professor L. Fox who provided wise counsel throughout the planning of the conference.

D. J. EVANS

Loughborough, December 1973.

Contents

1. Theory of Optimal Algorithms†

J. F. TRAUB

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa., U.S.A.

1. Introduction

Recent progress in the theory of optimal algorithms has led to new algorithms as well as theoretical bounds on the efficiency of any possible algorithm.

Historically there have been three major stages in the development of algorithmic analysis. They are:

1. Synthesis of *an* algorithm
2. Analysis of *an* algorithm
3. Analysis of *a class* of algorithms.

Initially the emphasis was on the synthesis of an algorithm. The second stage commenced around 1947 with the very careful analysis of particular algorithms. Within the last 10–15 years people have been looking at classes of algorithms and trying to find the best. This trend has recently accelerated and there is now tremendous interest in analyzing classes of algorithms in terms of computational complexity.

There are many reasons for studying computational complexity of which the most important are:

1. Constructing "good" new algorithms.
2. Filtering out "bad" algorithms.
3. Creating a theory of algorithms which will establish theoretical limits on computation.

To discuss optimal algorithms we need a measure of *cost*. The measure used throughout this paper is the total number of arithmetic operations, $+, -, \times, \div$. Gentleman (1973) and Reddy (1973) have discussed some of the

† This research was supported in part by the National Science Foundation under Grant GJ32111 and the Office of Naval Research under Contract N0014-67-A-0314-0010, NR 044-422.

other components of the cost which might be included. Other properties of a numerical algorithm, such as stability and domain of convergence, are critical. Measures of cost deserve more refinement.

2. Algebraic and Analytic Computational Complexity

We want to distinguish between two types of algorithms. The dichotomy depends on the nature of the underlying mathematical problem. A mathematical problem can be finite or infinite. Examples of finite problems are matrix multiplication and polynomial evaluation. Examples of infinite problems are the solution of an elliptic partial differential equation and the calculation of a polynomial zero.

Optimality theory for finite problems will be referred to as *algebraic computational complexity*, optimality theory for infinite problems as *analytic computational complexity*. Some examples will be given of work from each domain.

3. Recent Results in Algebraic Computational Complexity

Borodin (1973) gives a survey of the enormous recent activity in algebraic complexity. We will confine ourselves to some very recent results which deal with one set of related problems.

The problems are:

1. *Polynomial multiplication.* Given two polynomials of degree n, to find the product polynomial.
2. *Polynomial division.* Given two polynomials of degree n and $\frac{1}{2}n$, to find their quotient and remainder. More generally we divide a polynomial of degree n by a polynomial of degree m. The choice of $m = \frac{1}{2}n$ makes the "size" of the problem depend on just one parameter.
3. *Polynomial interpolation.* Given (x_i, y_i), $i = 0,1, \ldots, n$. Find $P(t)$ such that $P(x_i) = y_i$.
4. *Evaluation of a polynomial at many points.* Evaluate an nth degree polynomial at $n + 1$ points given simultaneously.
5. *Evaluation of a polynomial and all its derivatives.* Evaluate an nth degree polynomial and all its derivatives at one point.

These problems take $O(n^2)$ operations classically. Using "fast" algorithms the first two problems can be done in $O(n \log n)$ operations while the next three problems can be done in $O(n \log^2 n)$ operations. Fast polynomial multiplication is done with the Fast Fourier Transform. Other fast algorithms are due to Moenck and Borodin (1972), Strassen (1973), and Kung (1973). Borodin (1973) summarizes the state of the art in fast algorithms.

The results above are asymptotic. They are only significant for rather large values of n. For example n^2 is smaller than $n \log^2 n$ until n is somewhat greater than 30. (All logarithms are to base 2.) Furthermore, analyses ignore asymptotic constants which can prove significant if n is not too large (Borodin, 1973).

The following is an example of a new algorithm which is better than the best previously known algorithm, not just asymptotically, but for all n. Given

$$P(t) = \sum_{j=0}^{n} a_{n-j} t^j,$$

and a number x, the problem is to calculate the normalized derivatives $P^{(j)}(x)/j!$, $j = 0, \ldots, n$. The standard algorithm is some 150 years old and appears in most numerical methods texts. It is known as the iterated Horner rule or a synthetic division. This algorithm can be written as

$$\begin{aligned}
T_i^{-1} &= a_{i+1}, \qquad i = 0, 1, \ldots, n-1,\\
T_j^j &= a_0, \qquad j = 0, 1, \ldots, n,\\
T_i^j &= T_{i-1}^{j-1} + xT_{i-1}^j, \qquad j = 0, 1, \ldots, n-1, i = j+1, \ldots, n.
\end{aligned}$$

It is not difficult to verify that

$$\frac{P^{(j)}(x)}{j!} = T_n^j, \qquad j = 0, 1, \ldots, n.$$

Observe that the first two lines of the algorithm define initial conditions. All the work is done in the recursion of the last line. The recursion is done $\frac{1}{2}n(n+1)$ times and there is one addition and one multiplication per step. Thus the iterated Horner algorithm requires $\frac{1}{2}n(n+1)$ multiplications and $\frac{1}{2}n(n+1)$ additions.

Consider now the following algorithm.

$$\begin{aligned}
T_i^{-1} &= a_{i+1}x^{n-i-1}, \qquad i = 0, 1, \ldots, n-1,\\
T_j^j &= a_0x^n, \qquad j = 0, 1, \ldots, n,\\
T_i^j &= T_{i-1}^{j-1} + T_{i-1}^j, \qquad j = 0, 1, \ldots, n-1, i = j+1, \ldots, n.
\end{aligned} \tag{3.1}$$

It may be shown (Shaw and Traub, 1974a) that

$$\frac{P^{(j)}(x)}{j!} = x^{-j}T_n^j, \qquad j = 0, 1, \ldots, n-1.$$

In this algorithm all the multiplications are done as part of the initial conditions. The recursion involves additions only. The normalized derivatives are obtained by division using the x^j calculated as part of the initialisation.

Thus this algorithm, which is just as simple as the iterated Horner rule, yields the normalized derivatives in $3n - 2$ multiplications and divisions and $\frac{1}{2}n(n + 1)$ additions. The algorithm is of practical utility. It is also of theoretical interest since it demonstrates that only a *linear* number of multiplications and divisions are needed.

The problem posed here is a special case of the problem of calculating m derivatives of an nth degree polynomial. The algorithm presented above is a member of a one-parameter family of algorithms (Shaw and Traub, 1974a). The optimal choice of the parameter as a function of m and n is discussed by Shaw and Traub (1974b). Stability of these algorithms is established by Wozniakowski (1973).

4. An Efficiency Measure

The remainder of this paper deals with analytic computational complexity. Recent research includes the complexity of elliptic partial differential equations (Eisenstat and Schultz, 1973) and the complexity of systems of non-linear equations (Brent, 1973). A more extensive bibliography may be found in Traub (1972).

We confine ourselves here to the problem of calculating a real simple zero α of a real function f. This zero-finding problem may seem rather specialised, but it is equivalent to the fixed-point problem, a ubiquitous problem in mathematics and applied mathematics. It may be formulated in an abstract setting and covers partial differential equations, integral equations, and many other important problems, Traub (1972) and Kung and Traub (1973a, 1973b) may be consulted for the results reported in the rest of this paper and for proofs of the theorems.

Consider iteration algorithms for approximating α. Let the x_i be generated by an iteration function ϕ,

$$x_{i+1} = \phi(x_i)$$

To define an efficiency measure for ϕ we need measures of *goodness* and *cost*. As the measure of goodness we use the order p defined as follows. If

$$\lim_{x_i \to \alpha} \frac{\phi(x_i) - \alpha}{(x_i - \alpha)^p} = S \neq 0$$

then $p = p(\phi)$ is the *order of convergence*.

The cost consists of two parts: the *evaluation cost* and the *combinatory cost*. Let ϕ use v_i evaluations of $f^{(i)}$. If $f^{(i)}$. If $f^{(i)}$ is rational, let $c(f^{(i)})$ denote the number of arithmetic operations for one evaluation of $f^{(i)}$;

otherwise let $c(f^{(i)})$ denote the number of arithmetic operations used in the rational subroutine which approximates $f^{(i)}$. Then

$$\text{Evaluation cost} = \sum_{i \geqslant 0} v_i \, c(f^{(i)}).$$

Let $a(\phi)$ be the minimum number of arithmetic operations to combine the $f^{(i)}$ to form ϕ by any procedure λ. Then

$$\text{Combinatory cost} = a(\phi).$$

Finally, the cost of performing one iteration step is

$$\sum_{i \geqslant 0} v_i \, c(f^{(i)}) + a(\phi).$$

We define the efficiency $e(\phi, f)$ of the iteration ϕ with respect to the problem f by

$$e(\phi, f) = \frac{\log p(\phi)}{\sum_{i \geqslant 0} v_i \, c(f^{(i)}) + a(\phi)}. \tag{4.1}$$

A discussion of this efficiency measure, including its relation to other efficiency measures, is given by Kung and Traub (1973b). Here I will only point out that earlier measures (Traub, 1972) did not include the combinatory cost $a(\phi)$) and that inclusion of combinatory cost is crucial.

The efficiency measure has the following two properties:

1. It is invariant under composition.
2. It is inversely proportional to total cost.

The first property can be written as

$$e(\phi \cdot \phi, f) = e(\phi, f),$$

where $\phi \cdot \phi$ denotes performing the iteration ϕ twice. This says that a sequence and a subsequence have the same efficiency. The second property is stated more precisely as follows. Let ϕ_1, ϕ_2 be two iterations used to approximate α to within a certain accuracy. Let the total cost of ϕ_j be W_j. Then

$$\frac{e(\phi_1, f)}{e(\phi_2, f)} \sim \frac{W_2}{W_1}.$$

Let

$$c_f = \min_{i \geqslant 0} c(f^{(i)}).$$

In this paper, we refer to c_f as the *problem complexity*. Let

$$v(\phi) = \sum_{i \geqslant 0} v_i \, (\phi).$$

Clearly, $v(\phi)$ is the total number of evaluations used in ϕ. Then by (4.1),

$$e(\phi, f) \leqslant \frac{\log p(\phi)}{v(\phi)\, c_f + a(\phi)}. \tag{4.2}$$

This will be useful for obtaining upper bounds for $e(\phi, f)$.

The optimal efficiency depends on the family Φ to which ϕ belongs. Our classification for ϕ depends on the information required by ϕ. We can distinguish between iterations with or without *memory*. We restrict ourselves here to iterations without memory. That is, the new iterate x_{i+1} is computed using information only at the current iterate x_i. For iterations without memory we distinguish between *one-point iteration* and *multipoint iteration*. Roughly speaking, if f or its derivaties require evaluation at k points in order to generate a new iterate by the iteration ϕ, then ϕ is a k-point iteration. In particular, if $k = 1$ we call ϕ a one-point iteration and if $k > 1$ and the value of k is not important we call ϕ a multipoint iteration. This terminology was introduced by Traub (1964). Precise definitions are given by Kung and Traub (1973a).

The following two examples illustrate the definitions.

Example 4.1. (Newton–Raphson Iteration)

$$\phi(f)(x) = x - \frac{f(x)}{f'(x)}.$$

This is a one-point iteration with $p(\phi) = 2$, $v_0(\phi) = v_1(\phi) = 1$, and $a(\phi) = 2$. Hence

$$e(\phi, f) = \frac{1}{c(f) + c(f') + 2},$$

$$e(\phi, f) \leqslant \frac{1}{2c_f + 2}.$$

Example 4.2

$$z_0 = x,$$

$$z_1 = z_0 - \frac{f(z_0)}{f'(z_0)},$$

$$\phi(f)(x) = z_1 - \frac{f(z_1)f(z_0)}{[f(z_1) - f(z_0)]^2}\,\frac{f(z_0)}{f'(z_0)}.$$

This is a two-point iteration with

$$p(\phi) = 4, v_0(\phi) = 2, \qquad v_1(\phi) = 1 \quad \text{and} \quad a(\phi) = 8.$$

Hence

$$e(\phi, f) = \frac{2}{2c(f) + c(f') + 8},$$

$$e(\phi, f) \leqslant \frac{2}{3c_f + 8}.$$

Given an algorithm ϕ and a problem f, we can use $e(\phi, f)$ as defined by (4.1) to calculate efficiency. We are also interested in the optimal efficiency of a class of algorithms. This motivates the following definitions.

It is natural to ask for a given problem f what is the optimal value of $e(\phi, f)$ for all ϕ belonging to some family Φ. Define

$$E_n(\Phi, f) = \sup_{\phi \in \Phi} \{e(\phi, f) \mid v(\phi) = n\}.$$

Thus $E_n(\Phi, f)$ is the optimal efficiency over all $\phi \in \Phi$ which use n evaluations. Define

$$E(\Phi, f) = \sup\{E_n(\Phi, f) \mid n = 1, 2, \ldots\}.$$

Thus $E(\Phi, f)$ is the optimal efficiency for all $\phi \in \Phi$. We will establish lower and upper bounds for $E_n(\Phi, f)$ and $E(\Phi, f)$ with respect to different families of iterations. When there is no ambiguity, we write E_n (Φ, f) and $E(\Phi, f)$ as $E_n(f)$ and $E(f)$, respectively. Since in practice we are more concerned with efficiency for problems f with higher complexity, we are particularly interested in the asymptotic behavior of these bounds as $c_f \to \infty$.

5. Efficiency of One-Point Iteration

The iterations most used in practice are one-point iterations. We derive lower and upper bounds on the efficiency of any one-point iteration.

We consider a particular family of one-point iterations $\{\gamma_n\}$. The first three members of this family are given by

$$\gamma_1 = x$$

$$\gamma_2 = \gamma_1 - \frac{f(x)}{f'(x)}$$

$$\gamma_3 = \gamma_2 - \frac{f''(x)}{2f'(x)} \left[\frac{f(x)}{f'(x)}\right]^2.$$

The family γ_n has been thoroughly studied (Traub, 1964, Section 5.1). Its important properties from our point of view are summarized in the following

THEOREM 5.1

1. $v_i(\gamma_n) = 1, \quad i = 0, 1, \ldots, n-1, v_i(\gamma_n) = 0, i > n-1.$
   Hence $v(\gamma_n) = n.$
2. $p(\gamma_n) = n.$

It can be shown (Kung and Traub (1973b)) that

$$a(\gamma_n) \leqslant \rho n^2 \log n \tag{5.1}$$

for some positive constant ρ. By (5.1) and Theorem 5.1,

$$e(\gamma_n, f) \geqslant \frac{\log n}{\sum_{i \geqslant 0}^{n-1} c(f^{(i)}) + \rho n^2 \log n}. \tag{5.2}$$

For n small, $a(\gamma_n)$ can be calculated by inspection. Thus $a(\gamma_3) = 7$ and

$$e(\gamma_3, f) = \frac{\log 3}{c(f) + c(f') + c(f'') + 7}.$$

We now turn to general one-point iterations. Let ϕ be any one-point iteration, with $v(\phi) = n$, which satisfies a mild smoothness condition. Then by Traub (1964, Section 5.4), Kung and Traub (1973b, Theorem 6.1), $v_i(\phi) \geqslant 1$, $i = 0, \ldots, p(\phi) - 1$ and hence $p(\Phi) \leqslant n$. Since at least $n - 1$ arithmetic operations are needed to combine n evaluations of f and its derivatives, $a(\gamma_n) \geqslant n - 1$.

Hence, from (4.2),

$$e(\phi, f) \leqslant \frac{\log n}{nc_f + n - 1} = h(n). \tag{5.4}$$

It may be verified that

$$h(n) \leqslant h(3) = \frac{\log 3}{3c_f + 2}, \quad \text{for all } n, \text{for } c_f > 4. \tag{5.5}$$

Since it is important to solve "difficult" problems efficiently, the condition $c_f > 4$ is not restrictive. Since

$$h(2) = \frac{\log 2}{2c \quad + 1} = \frac{1}{2c \quad + 1}$$