NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

CMU-CS-71-105

COMPUTATIONAL COMPLEXITY OF ITERATIVE PROCESSES

J. F. Traub

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa.

October 1971

This research was supported in part by the Office of Naval Research under contract N 00014-67-A-0314-0010, NR 044-422 and by the Advanced Research Projects Agency under contract F44620-70-C-0107.

ABSTRACT

The theory of optimal algorithmic processes is part of computational complexity. This paper deals with analytic computational complexity. The relation between the goodness of an iteration algorithm and its new function evaluation and memory requirements are analyzed. A new conjecture is stated.

1. INTRODUCTION

Computational complexity is one of the foundations of theoretical computer science. The phrase computational complexity seems to have been first used by Hartmanis and Stearns [10] in 1965 although the first papers belonging to the field are those of Rabin [25, 26] in 1959 and 1960.

One of its important components is the theory of optimal algorithmic processes. We distinguish between optimality theory for algebraic (or combinatorial) processes, which we call algebraic computational complexity and optimality theory for analytic (or continuous) processes, which we call analytic computational complexity.

The last few years have witnessed striking developments in algebraic computational complexity; for example, the multiplication of numbers (Cook [5], Schönhage and Strassen [28]), the multiplication of matrices (Winograd [38], Strassen [29], Hopcroft and Kerr [12]), polynomial evaluation (Winograd [38]), median of a set of numbers (Floyd [9]), graph planarity (Hopcroft and Tarjan [13]). Surveys may be found in Knuth [17], Borodin [1], and Minsky [21].

Research on analytic computational complexity dates to the early sixties (Traub [30-36] and predates most of the algebraic results. More specifically the work on analytic computational complexity to date has concerned optimal iteration. Recent results are due to Cohen [2], Cohen and Varaiya [3], Feldstein [6], Feldstein and Firestone [7, 8], Hindmarsh [11], Jarratt [14], King [16], Miller [19, 20], Paterson [24], Rissanen [27], and Winograd and Wolfe [39]. (Paterson's results are summarized at the end of Section 2.)

In this paper we define basic concepts and pose some fundamental questions in optimal iteration. In the terminology of Knuth [18] we perform a Type B analysis. That is, we consider a family of algorithms for solving a particular problem and select the "best possible". We survey earlier work, report recent progress, and state a new conjecture. Since the field is changing rapidly, some of the results cited have not yet appeared in the open literature. An abbreviated version of this material was presented (Traub [37]) at the IFIP 71 Congress, with somewhat different terminology and notation.

This paper is intended for the non-specialist in iteration theory and therefore some precision in definitions and some generality in the models of iteration algorithms are sacrificed.

2. BASIC CONCEPTS

We begin by specifying the problem. Let F denote the class of infinitely differentiable real functions defined on the real line. We assume that if $f \in F$, then f has at least one simple zero α, that is, a number α such that $f(\alpha) = 0$, $f'(\alpha) \neq 0$. The assumption of infinite differentiability is for simplicity. For any algorithm we shall discuss, f need only have a small number of derivatives on a finite interval.

Our problem is to approximate α for $f \in F$. This zero-finding problem may seem rather specialized, but in fact, it is equivalent to the fixed-point problem of calculating a number α such that $\alpha = g(\alpha)$, an ubiquitous problem in mathematics and applied mathematics. It may be formulated in an abstract setting and covers partial differential equations, integral equations, boundary value problems for ordinary differential equations as well as many other important problems (Collatz [4]).

We consider iterative algorithms for the approximation of α. A sequence of approximating iterates $\{x_i\}$ is generated by an iteration function. We shall not give a formal definition of iteration algorithm. The interested reader may consult Ortega and Rheinboldt [22] and Cohen and Varaiya [3].

Our aim is to discuss optimal iteration algorithms. There are a number of measures we could optimize. For example, we could minimize the total number of arithmetic operations needed to approximate α to within an error ϵ. This measure is strongly dependent on the particular f in question. For our current purpose, we prefer a measure which is not so dependent on f

and which is easier to calculate. (At the end of this section we report a recent optimality result which does optimize arithmetic operations.)

We introduce general measures of cost and goodness. The cost consists of two parts: the new evaluation cost e and the memory cost m. The new evaluation cost e is defined as the number of new function evaluations required. This definition is motivated by the following considerations. An iteration step consists of two parts.

1. Calculate new function values.
2. Combine the data to calculate the next iterate.

Since the evaluation of functions requires invocation of subroutines whereas the calculation of the next iterate requires only a few arithmetic operations, we neglect the latter.

A function evaluation is the calculation of f or one of its derivatives. Thus if $f(x_i)$ and $f'(x_i)$ are required, e = 2. We could assign a new evaluation cost of θ_j for the evaluation of $f^{(j)}$ (Traub [36, p. 262]), but this would make the measure f-dependent.

We turn to the second component of the cost. If previous function evaluations at $x_{i-1}, \ldots, x_{i-m}$ are used to calculate x_{i+1}, then we define m as the memory cost of the iteration.

Another component of the cost is not included in this paper. An iteration such as the secant iteration involves the subtraction of quantities which are close together, and to maintain accuracy, more precision should perhaps be carried. The theory should be extended to include this cost.

We turn now to a measure of the goodness of an iteration. Let $x_i \to \alpha$. If there exists a number p such that

$$\lim_{x_i \to \alpha} \frac{|x_{i+1} - \alpha|}{|x_i - \alpha|^p} = A \neq 0, \infty$$

then p is called the order of the iteration. This definition of order will serve for our purposes. For other definitions of order the reader is referred to Ortega and Rheinboldt [22] and Cohen and Varaiya [3].

This is a reasonable measure of goodness since if A is near unity, then x_{i+1} has about p times as many significant figures as x_i.

The order has two additional properties which make it useful for our purposes. It depends primarily on the algorithm and only weakly on f and it is fairly easy to calculate. For example, for all twice continuously differentiable functions f for which $f''(\alpha) \neq 0$, Newton iteration (see Example 1 below) has order $p = 2$. (Recall we are assuming throughout this paper that $f'(\alpha) \neq 0$.) Under the same conditions, the secant iteration has order $p = \frac{1}{2}(1 + \sqrt{5}) \doteq 1.62$.

Two widely known iteration algorithms may serve to illuminate these definitions. We will use them to introduce data flow charts which are a convenient way to describe algorithms from our point of view.

Example 1. Newton Iteration. Let x_i be given. Define

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = \Phi[x_i, f(x_i), f'(x_i)].$$

The data flow chart of Figure 1 exhibits the process at step i. For Newton iteration,

$$e = 2$$
$$m = 0$$
$$p = 2 \quad (\text{if } f''(\alpha) \neq 0).$$

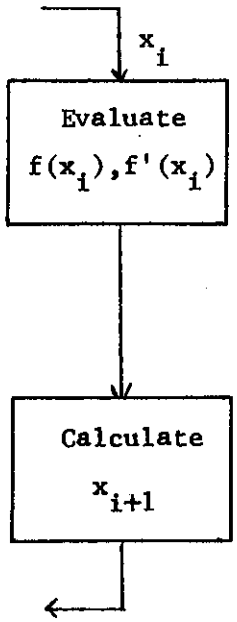

FIGURE 1. DATA FLOW CHART FOR NEWTON ITERATION

Example 2. Secant Iteration. Let x_0, x, be given. Define

$$x_{i+1} = x_i - f(x_i)\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

$$= \Phi[x_i, x_{i-1}, f(x_i), f(x_{i-1})].$$

The data flow chart of Figure 2 exhibits the process at step i. For secant iteration,

$$e = 1$$
$$m = 1$$
$$p = \frac{1}{2}(1 + \sqrt{5}) \doteq 1.62 \quad (\text{if } f''(\alpha) \neq 0).$$

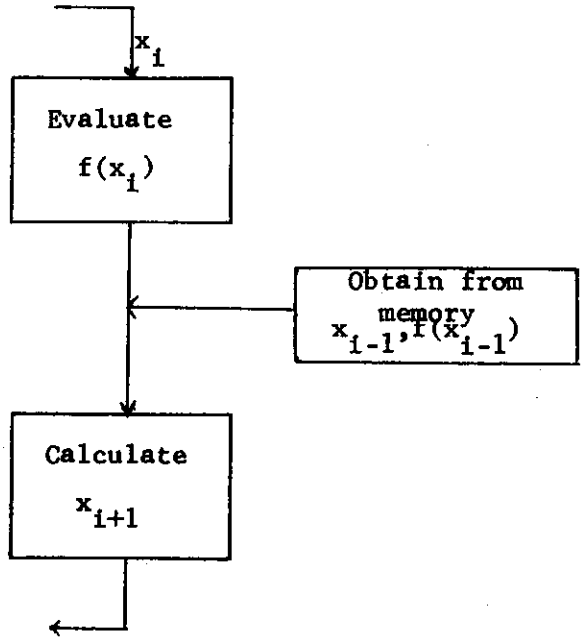

FIGURE 2. DATA FLOW CHART FOR SECANT ITERATION

We now pose the following optimality questions which will be our focus for the remainder of this paper. Other optimality problems will be discussed at the end of this section.

TWO OPTIMALITY QUESTIONS

1. What is the maximal order $P_{e,m}$ which can be achieved for iterations which use e new function evaluations and have memory m?

2. What is the most that memory m is worth? That is, what is $P_{e,m} - P_{e,o}$?

The answers depend on the class of iterations under study. Traub [36, Section 1.22] introduced four classes depending on the function evaluation and memory requirements of the algorithms. These classes are:

One - Point
One - Point With Memory
Multipoint
Multipoint With Memory

We shall discuss optimality results for only the first three classes in this paper.

These classes model algorithms appropriate for stationary iterations on sequential machines. An iteration rule is stationary if it does not change from step to step. A formal definition may be found in Ortega and Rheinboldt [22]. Because of the assumption of sequential machine, the definition of one point iteration with memory (Section 5) uses the same number of derivatives at each point. On parallel machines we may want to vary the number of derivatives at each point. The case where the number of derivatives varies is studied by Traub [36, pp. 60-65] and Feldstein and Firestone [7].

Besides those posed earlier, we discuss some additional optimality questions. An important measure of the goodness of an algorithm is the efficiency index defined by

$$E = p^{\frac{1}{e}}$$

(See Ostrowski [23, Chapter 3] and Traub [36, Appendix C] for discussion of this index.) A study of iterations with high values of the efficiency index is reported by Feldstein and Firestone [7].

An efficiency index similar to E has been used by Paterson [24] to derive a most interesting result concerning the calculation of square roots. The calculation of $\sqrt{A}$ is identical with calculating the positive zero of $f = x^2 - A$. Paterson takes for his efficiency measure of the iteration ϕ,

$$V(\phi) = V = \frac{\log_2 p}{M}$$

where p denotes the order and M denotes the number of multiplications or divisions (except by constants). Since Paterson deals with a particular f, M is a good measure. Note that $V = \log_2 E$ with M replacing e as a measure of cost.

Paterson observes that for Newton iteration, $V = 1$. He proves

THEOREM (Paterson [24])

<u>If ϕ is a rational functional with rational coefficients, generating a sequence converging to an algebraic number α and with order greater than unity, then $V \leq 1$.</u>

Thus Newton iteration is optimal for the calculation of square roots, at least among iterations with rational coefficients.

It is part of the folklore of numerical mathematics that it is better to do something simple more times then something more complicated fewer times. Paterson's result may be interpreted as stating and proving this rigorously for a particular problem.

3. INTERPOLATORY ITERATION

Before discussing optimality results for classes of iterations, we discuss particular families of iterations which play a special role in the theory, the interpolatory iteration algorithms $I_{e,m}$ introduced and analyzed by Traub [35], [36]. For our purpose here, we need not know how formulas for interpolatory iteration are derived. Indeed, there are two families of interpolatory iterations derived from direct and inverse iteration. Both families have the same order for a given e and m and we shall not distinguish between them. In both families, $I_{2,0}$ is Newton iteration and $I_{1,1}$ is secant iteration.

For interpolatory iterations we have a complete theory relating order to evaluation and memory costs. Let $q_{e,m}$ denote the order of an interpolatory iteration $I_{e,m}$. Then we have the following basic result.

THEOREM (Traub [36, Section 3.3 and 6.1])

$q_{e,0} = e$. For all finite e and $m > 0$, $e < q_{e,m} < e+1$. For e fixed, $q_{e,m}$ is a strictly increasing function of m and

$$\lim_{m \to \infty} q_{e,m} = e+1.$$

This is a very satisfying result. It says that for interpolatory iteration, increasing memory while keeping the number of new evaluations fixed always increases the order.

An important corollary is

COROLLARY (Traub [36, Section 6.1])

For all finite m, $q_{e,m} - q_{e,0} < 1$.

Thus for interpolatory iterations memory adds less than unity to the order.

Upper and lower bounds on the order are given by the following theorems. Let

$$\delta_{e,m} = e+1 - q_{e,m}$$

and let e denote the base of natural logarithms.

THEOREM (Traub [36, Section 3.3])

$$\frac{e}{(e+1)^m} < \delta_{e,m} < \frac{ee}{(e+1)^m}$$

A sharper result is given by

THEOREM (Kahan [15])

$$\frac{e}{(e+1)^{m+1} - 1} \leq \delta_{e,m} \leq \frac{e}{(e+1)^{m+1} - 1 - \frac{em}{e+1}}$$

Values of $q_{e,m}$ for small values of e and m may be found in Table 1.

	e = 1	e = 2	e = 3
m = 0	1.000	2.000	3.000
m = 1	1.618	2.732	3.791
m = 2	1.839	2.920	3.951
m = 3	1.928	2.974	3.988

TABLE 1. VALUES OF $q_{e,m}$

4. ONE-POINT ITERATION

An iteration function belongs to the class of one-point iterations if all new function evaluations are at the point x_i and if its memory $m = 0$. Thus

$$x_{i+1} = \Phi_{e,0}[x_i, f(x_i), \ldots, f^{(e-1)}(x_i)].$$

The data flow chart for a one-point iteration is given in Figure 3.

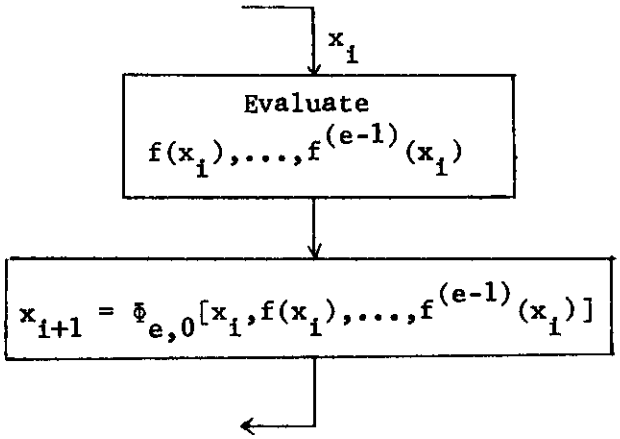

FIGURE 3. DATA FLOW CHART FOR ONE-POINT ITERATION

For one-point iterations the first optimality question is settled by the theorem below. Recall that $P_{e,m}$ is the optimal order for an iteration characterized by new function evaluations e and memory m.

THEOREM (Traub [30, 36, Section 5.4])

$$P_{e,0} = e.$$

5. ONE-POINT ITERATION WITH MEMORY

An iteration function belongs to the class of one-point iterations with memory if all new function evaluations are at the point x_i and if its memory $m > 0$. Thus

$$x_{i+1} = \Phi_{e,m}[x_i, f(x_i), \ldots, f^{(e-1)}(x_i);$$

$$x_{i-1}, f(x_{i-1}), \ldots, f^{(e-1)}(x_{i-1}),$$

$$x_{i-m}, f(x_{i-m}), \ldots, f^{e-1)}(x_{i-m})].$$

The semi-colon separates new function evaluations from those recovered from memory. The data flow chart for a one-point iteration with memory is given in Figure 4.

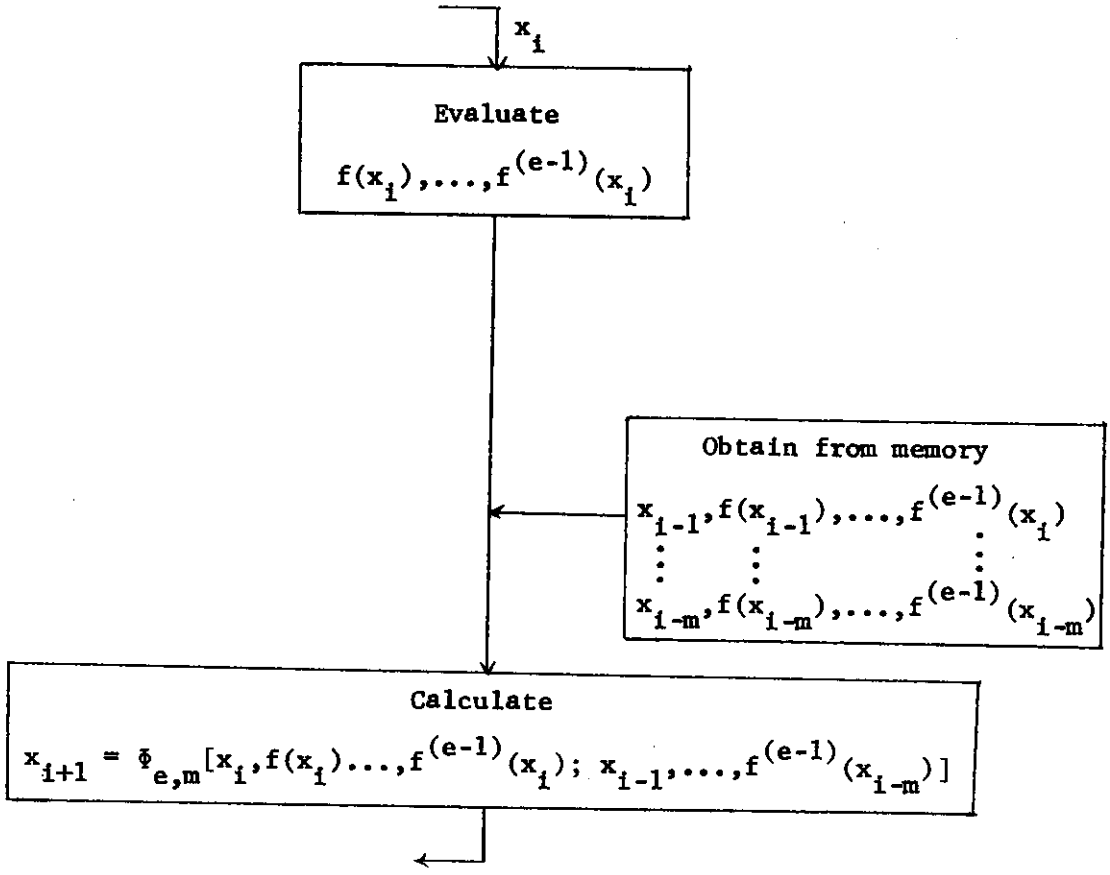

FIGURE 4. DATA FLOW CHART FOR ONE-POINT ITERATION WITH MEMORY

The initial conjecture on optimality for this class was reported at the 1961 National ACM Conference.

CONJECTURE (Traub [30, 36 (Section 6.3)])

For all one-point iterations with finite memory m

$$P_{e,m} - P_{e,0} < 1.$$

Until the late sixties no progress was reported, but there have been exciting recent results. The matter has been investigated by Winograd and Wolfe [39] who assert a stronger result. Under weak conditions on the admissible iteration functions, interpolatory iteration $I_{e,m}$ is optimal among all iterations characterized by new function evaluations e and memory m. The truth of the conjecture then follows from the Corollary in Section 3.

Winograd and Wolfe [39] have pointed out an ambiguity in the notion of memory since instead of using memory explicitly at each step, one can use it implicitly by encoding it in other data. Cohen and Varaiya [3] cite an example of such an encoding. Cohen and Varaiya deal with the ambiguity by adding a condition to the definition of order which insures that encoding does not increase the rate.

Rissanen [27] resolves the ambiguity by imposing a smoothness condition on admissible algorithms. He proves that then the secant iteration (that is, the interpolatory iteration $I_{1,1}$ has maximal order among all algorithms one with e = 1, m = 1.

6. MULTIPOINT ITERATION

We summarize the situation for one-point iterations with or without memory. A one-point iteration with e new function evaluations (and therefore e-1 derivatives) is of order at most e. A one-point iteration with memory with e new function evaluations (and therefore e-1 derivatives) is of order less than e+1. Table 2 summarizes the situation.

	New function evaluations	Highest derivative	Optimal Order
One-Point	e	e-1	e
One-Point With Memory	e	e-1	$< e+1$

TABLE 2. SUMMARY OF FACTS ABOUT ITERATION FUNCTIONS

Is there a class of iteration algorithms for which these restrictions do not hold? An affirmative answer is provided by multipoint iterations (Traub [34], [36, Section 1.2]).

An iteration function belongs to the class of multipoint iterations if new function evaluations are made at more than one point and if its memory $m = 0$.

We shall confine ourselves to giving a general prescription and a data flow chart of a multipoint iteration only for the case of a two-point iteration. Then

$$z_i = \phi[x_i, f(x_i), \ldots, f^{(e_1-1)}(x_i)]$$

$$x_{i+1} = \psi[x_i, f(x_i), \ldots, f^{(e_1-1)}(x_i), z_i, f(z_i), \ldots, f^{(e_2-1)}(z_i)].$$

The data flow chart is given by Figure 5.

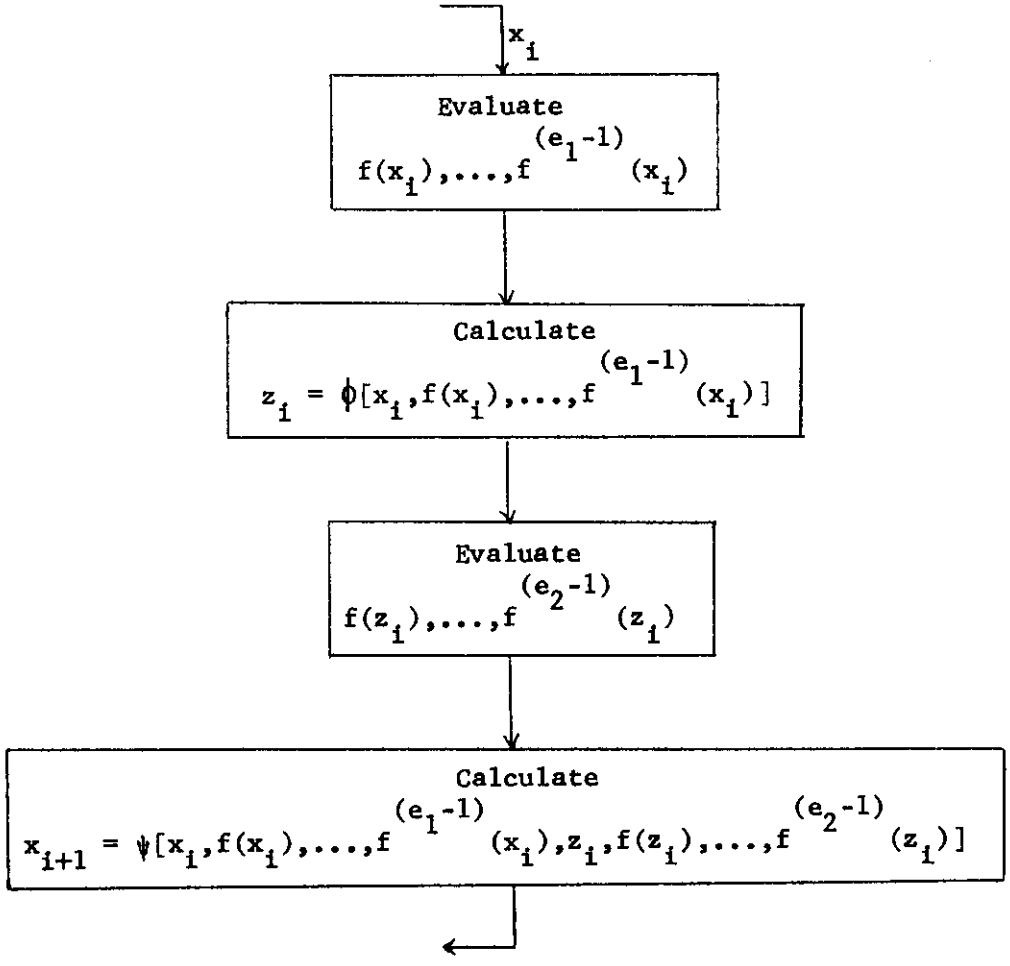

FIGURE 5. DATA FLOW CHART FOR TWO-POINT ITERATION

A fourth class of iterations, multipoint with memory, is defined by Traub [36, Section 1.2]. We shall not discuss multipoint iteration with memory here.

Table 2 lists two types of requirements, one on the total number of new function evaluations and a second on the highest derivative required. First we give examples to show that the restriction on derivatives need not apply for multipoint iterations.

Example 1

$$x_{i+1} = \psi(x_i) = \frac{x_i\phi(\phi(x_i)) - \phi^2(x_i)}{x_i - 2\phi(x_i) + \phi(\phi(x_i))},$$

$$\phi(x_i) = x_i - f(x_i)$$

This is a particular case of the Steffensen-Householder-Ostrowski iteration (Traub [36, Appendix D]). Note that no derivatives are used. Yet if $f'(\alpha) \neq 1$, $p = 2$.

Example 2

Let $L \geq 3$ be fixed and let

$$x_{i+1} = \phi[x_i, f(x_i), f'(x_i), f(\lambda_2), \ldots, f(\lambda_{L-1})],$$

where

$$\lambda_j = \lambda_j(x_i) = \lambda_{j-1}(x_i) - \frac{f(\lambda_{j-1}(x_i))}{f'(x_i)}, \quad j=2,\ldots,L-1$$

and

$$\lambda_1(x_i) = x_i.$$

This is a multipoint iteration based on L-1 points. The new function evaluations are L-1 evaluations of f and one of f'. For all twice continuously differentiable f for which $f''(\alpha) \neq 0$, this iteration is of order L (Traub [36, Section 8.34].

These two examples show that for multipoint iterations there is no connection between the highest derivative required and the order.

For these two examples, the order equals the number of new function evaluations. Since we proved this was always the case for one-point iterations, we might be tempted to suppose that this result holds for multipoint iterations also. That this is not the case is shown by the following example.

Example 3

$$z_i = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}\left[\frac{f(z_i) - f(x_i)}{2f(z_i) - f(x_i)}\right]$$

The data flow chart is given in Figure 6.

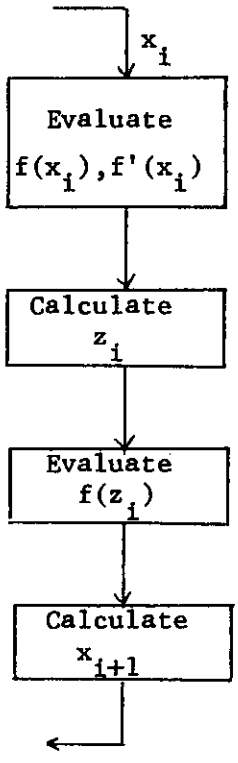

FIGURE 6. AN EXAMPLE OF A MULTIPOINT ITERATION

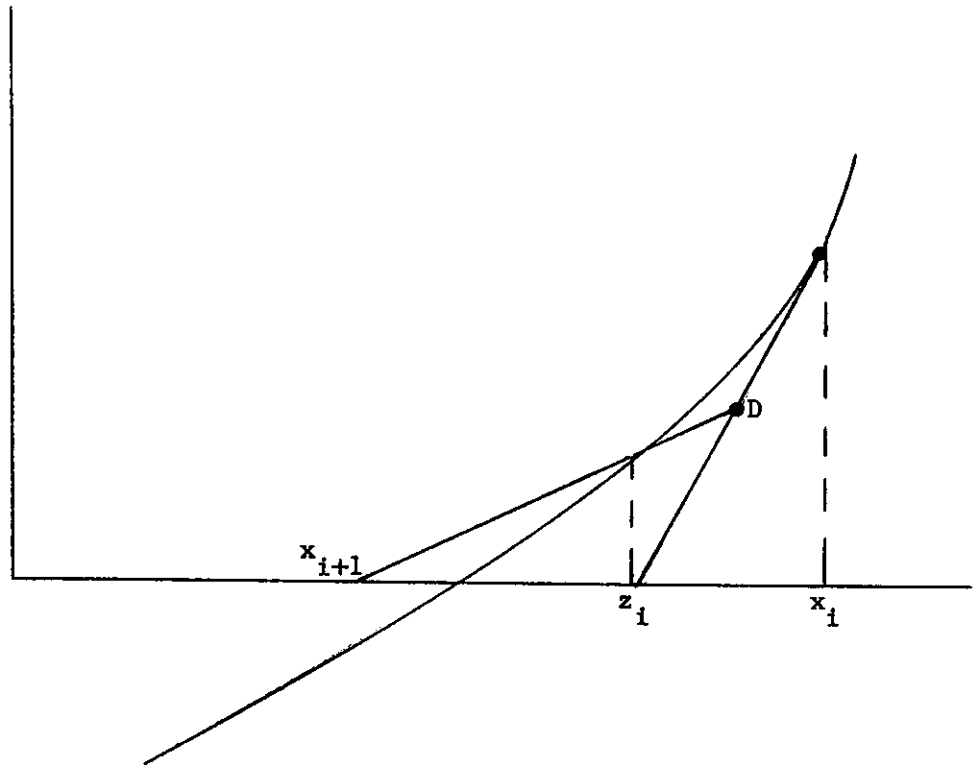

FIGURE 7. GEOMETRIC INTERPRETATION
D is midpoint of line between (z_i, o) and $(x_i, f(x_i))$

This iteration uses two evaluations of f and one of f'. Jarratt [14] has constructed a fourth order iteration using just two evaluations of f' and one of f. King [16] constructs a family of fourth order methods which use two values of f and one value of f'.

We turn to optimality considerations for multipoint iterations. As before let $P_{e,0}$ denote the maximal order for an iteration with new function evaluations e and no memory. If we permit only one-point or multipoint iterations (no memory), we know that $P_{2,0} \geq 2$ (Newton iteration) and $P_{3,0} \geq 4$ (Example 3 above).

We state a

NEW CONJECTURE

For all one-point or multipoint iterations without memory,

$$P_{2,0} = 2$$
$$P_{3,0} = 4$$

BIBLIOGRAPHY

[1] Borodin, A., Computational Complexity - A Survey. Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems (1970), 257-262.

[2] Cohen, A. I., Rate of Convergence and Optimality Conditions of Root Finding and Optimization Algorithms, University of California, Berkeley Dissertation, 1970.

[3] Cohen, A. I. and Varaiya, P., Rate of Convergence and Optimality Conditions of Root Finding. To appear.

[4] Collatz, L., Functional Analysis and Numerical Mathematics. Academic Press, 1964.

[5] Cook, S. A., On the Minimum Computation Time for Multiplication. Harvard Dissertation, 1966.

[6] Feldstein, A., Bounds on Order and Ostrowski Efficiency for Interpolatory Iteration Algorithms. University of California, Lawrence Radiation Laboratory, Livermore, 1969.

[7] Feldstein, A. and Firestone, R. M., Hermite Interpolatory Iteration Theory and Parallel Numerical Theory. Report, Division of Applied Mathematics, Brown University, 1967.

[8] Feldstein, A. and Firestone, R. M., A Study of Ostrowski Efficiency for Composite Iteration Functions. Proceedings ACM National Conference (1969), 147-155.

[9] Floyd, R. W., Notes on Computing Medians and Percentiles. To appear.

[10] Hartmanis, J. and Stearns, R. E., Computational Complexity of Recursive Sequences. IEEE Proceedings Fifth Annual Symposium on Switching Circuit Theory and Logical Design (1964), 82-90.

[11] Hindmarsh, A. C., Optimality in a Class of Rootfinding Algorithms. University of California, Lawrence Radiation Laboratory, Livermore Report UCRL-72456. To appear in SIAM Journal of Numerical Analysis.

[12] Hopcroft, J. E. and Kerr, L. E., On Minimizing the Number of Multiplications Necessary for Matrix Multiplication. SIAM Journal of Applied Mathematics 20 (1971), 30-36.

[13] Hopcroft, J. and Tarjan, R., Planarity Testing in V log V steps: Extended Abstract. Proceedings IFIP Congress Booklet TA-2 (1971), 18-22.

[14] Jarratt, P., Some Efficient Fourth Order Multipoint Methods for Solving Equations. BIT 9 (1969), 119-124.

[15] Kahan, W., Private Communication, 1969.

[16] King, R. F., A Family of Fourth-Order Methods for Nonlinear Equations. To appear.

[17] Knuth, D., The Art of Computer Programming, Volume 2, Seminumerical Algorithms, Addison-Wesley, 1969.

[18] Knuth, D., Mathematical Analysis of Algorithms. Proceedings IFIP Congress Booklet I (1971), 135-143.

[19] Miller, W., Toward Abstract Numerical Analysis, University of Washington Dissertation, 1969.

[20] Miller, W., Unsolvable Problems With Differentiability Hypotheses. Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems (1970), 480-482.

[21] Minsky, M., Form and Computer Science. Journal ACM 17 (1970), 197-215.

[22] Ortega, J. M. and Rheinboldt, W. C., Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, 1970.

[23] Ostrowski, A. M., Solution of Equations and Systems of Equations, second edition. Academic Press, 1966.

[24] Paterson, M. S., Optimality for Square Root Algorithms. Private Communication, 1971.

[25] Rabin, M. O., Speed of Computation of Functions and Classification of Recursive Sets. Proceedings Third Convention of Scientific Societies, Israel, 1-2, 1959.

[26] Rabin, M. O., Degrees of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets. Technical Report 2, Hebrew University, Jerusalem, 1960.

[27] Rissanen, J., On Optimum Root-Finding Algorithms. IBM Report RJ726 (No. 13830).

[28] Schönhage, A. and Strassen, V., Schnelle Multiplication Grosser Zahlen, 1970. To appear.

[29] Strassen, V., Gaussian Elimination is Not Optimal, Numerische Math. 13 (1969), 354-356.

[30] Traub, J. F., On Functional Iteration and the Calculation of Roots. Proceedings 16th National ACM Conference, 5A-1 (1961), 1-4.

[31] Traub, J. F., On Functional Iteration and the Calculation of Roots. Unpublished Report (142 pages), 1961.

[32] Traub, J. F., Optimal m-Invariant Iteration Functions. Notices American Mathematical Society 9, No. 2 (1962), 122.

[33] Traub, J. F., On the Informational Efficiency of Iteration Functions. International Congress of Mathematicians, 1962.

[34] Traub, J. F., The Theory of Multipoint Iteration Functions. Proceedings 17th National ACM Conference (1962), 80-81.

[35] Traub, J. F., Interpolatively Generated Iteration Functions. Proceedings 18th National ACM Conference, 1963.

[36] Traub, J. F., Iterative Methods for the Solution of Equations. Prentice-Hall, 1964.

[37] Traub, J. F., Optimal Iterative Processes: Theorems and Conjectures. Proceedings IFIP Congress Booklet TA-1 (1971), 29-32.

[38] Winograd, S., The Number of Multiplications Involved in Computing Certain Functions. Proceedings IFIP Congress, Booklet A (1968), A-128-A130.

[39] Winograd, S. and Wolfe, P., Private Communication, 1969.

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
Carnegie-Mellon University
Pittsburgh, Pa.

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

COMPUTATIONAL COMPLEXITY OF ITERATIVE PROCESSES

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Scientific report

5. AUTHOR(S) *(First name, middle initial, last name)*

J. F. Traub

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October, 1971	27	39

8a. CONTRACT OR GRANT NO.
N 00014-67-A-0314-0010, NR 044-422
F44620-70-C-0107

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

CMU-CS-71-105

10. DISTRIBUTION STATEMENT

Approved for public release, distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Office of Naval Research
Advanced Research Projects Agency

13. ABSTRACT

Key words and phrases: computational complexity, optimal algorithm, optimal iteration, numerical mathematics, iteration theory.

The theory of optimal algorithmic processes is part of computational complexity. This paper deals with analytic computational complexity. The relation between the goodness of an iteration algorithm and its new function evaluation and memory requirements are analyzed. A new conjecture is stated.

DD FORM 1 NOV 68 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT