A survey of techniques in applied computational complexity (*)

Robert Meersman (**)

ABSTRACT

An attempt is made to introduce the non-expert reader to the many aspects of a relatively new and varied field which seems to be at the same time analysis, algebra and computer science. Computational complexity can be roughly described as the theory of optimizing finite and infinite algorithms for use on digital computers. Even for "simple" problems like the finding of a zero of a real function or even the evaluation of a polynomial, surprisingly deep techniques are necessary. A representative sample of the presently existing bibliography on the subject is included at the end.

1. INTRODUCTION

Programmers tend to become slightly overawed with the truly enormous speed and data handling capacity of modern day digital computers. This often results in clumsy and wasteful programming. In recent years however the use of such devices has greatly intensified, and consequently the demand for more efficient programs has risen, not to speak of situations where actual speed is essential in running certain programs. The search for "optimal" algorithms for specific problems implies the finding of good criteria for optimality or efficiency, and the need of classifying algorithms with respect to these criteria. This essentially constitutes a vast field of research currently labeled as computational complexity.

Part of this is taken up by "abstract" complexity theory, from the viewpoint of recursive function theory and the theory of Turing machine computations.

We will not be concerned here with this "general" aspect of computation; for an excellent survey the interested reader is referred to the papers of Hartmanis and Hopcroft [8] and of Borodin [1]. As even a theoretically "good" algorithm can be badly implemented, the study of the behaviour of a particular algorithm in a particular computer could also be considered part of complexity theory. This is what Knuth [14] calls a type A analysis. For a (very complete) treatment of this problem, see of course his books [15] [16] [17].

How difficult or complex is a particular problem ? Assume we have found a class of solutions for it, such that with each solution we can associate a real number called its complexity. We could then define the complexity of this problem relative to that class of algorithms as the infimum of these numbers. Naturally, chosen algorithms depend on the kind of complexity measure (Trivial example : for each element in one class of programs for the same problem, consider its size (s) and its execution time (t) as complexity measures. By increasing s, for instance through lookup tables, it is often possible to reduce (t). As a striking illustration of the fact that the oldest algorithms did not necessarily survive time because of their efficiency, consider the multiplication of two n-bit numbers. This is done classically in $0(n^2)$ "elementary" operations. However there exists a simple way to do it in $0(n^{\log 3})$ operations (see Knuth [16]) and in 1970 Schönhage and Strassen [41] discovered a method requiring only $0(n \log n \log \log n)$ steps, using the so-called Fast Fourier Transform and modular arithmetic (*).

This survey is divided into two parts, respectively treating - in the terminology of Traub [54] [56] - algebraic computational complexity (sections 2 to 5) and analytic computational complexity (sections 6 to 8). The first kind, loosely stated, treats problems of finite nature like the evaluation of polynomials, while the second is concerned with problems like for example the finding of a zero of a certain function. Our aim is to make the reader aware of the very wide variety of techniques presently in use in this field.

2. EVALUATION OF CERTAIN FUNCTIONS

Problem 1

To evaluate the polynomial $P_n(x)=a_n x^n+\ldots+a_1x+a_0$.

It is well known that this can be done in a simple way by the so called "Horner's Rule", i.e.

set $T_n = a_n$;

compute $T_i = T_{i+1} x + a_i$ for $i = n-1, n-2, \ldots, 1, 0$;

(*) all logarithms in this paper are to base 2.

(*) This paper is an extension of two talks given in Nov. 1973 at the Seminar on Numerical Analysis at the Universitaire Instelling Antwerpen

(**) Vrije Universiteit Brussel, Departement voor Wiskunde, Terhulpesteenweg 166, 1170 Brussel

then $P_n(x) = T_0$.

The complexity of such schemes is most often expressed in the number of arithmetic operations used. Clearly Horner's scheme uses n multiplications (MUL) and n additions/subtractions (ADD). It is difficult to imagine a cheaper way to do it, yet the optimality of Horner's Rule was not proven until 1958 by Pan [34]. Indeed we have the following

THEOREM

Among all algorithms with input $(a_0, a_1, \ldots, a_n; x)$ and output $P_n(x)$ Horner's scheme is optimal, i. e. it minimizes the number of Multiplications, Divisions and Additions, Subtractions.

(Remark : We leave it to the reader to construct an example where a polynomial can be evaluated in a cheaper way using division).

For a relatively simple proof, see Reingold and Stocks [38] : it proceeds by induction on the count of "crucial steps" (those steps of the algorithm which contribute to the operation count) by a substitution argument on the first crucial step.

It is essential to note that, in the theorem above, the coefficients of P_n are part of the input. Indeed, often the problem is to evaluate the same polynomial P_n at a large number of points (take for instance a library program which uses a polynomial to approximate a given function). In such a case it could be interesting to do some preliminary arithmetic on the coefficients in order to save multiplications at evaluation time. (In many practical cases it is a reasonable assumption to consider addition time negligible relative to multiplication or division time, e. g. when n is a multiple precision number or a square matrix). This procedure is commonly called preconditioning and seems to be introduced by Motzkin [29] and Pan [34].

Example : For any fourth degree polynomial P_4. It is possible (see for instance Knuth [1 §]) to determine $\alpha_0, \ldots, \alpha_4$ such that

$$P_4(x) = ((y + x + \alpha_2)y + \alpha_3)\alpha_4$$

where $y = (x + \alpha_0)x + \alpha_1$

and thus computing $P_4(x)$ with 3 MUL and 5 ADD. The result is to cut the number of necessary multiplications roughly in half; still we need at least

$$[\frac{1}{2}(n+1)] \text{ MUL and } n \text{ ADD}$$

(Winograd [60]; Winograd and Rabin [37]. In this paper, an algorithm using $\frac{1}{2}n + 0(\log n)$ MUL and $n + 0(n)$ ADD can be found). For some simple proofs see Reingold and Stocks [38].

Some results are also beginning to be available on the simultaneous evaluation of an n^{th} degree polynomial at $n+1$ arbitrary points. This is "dual" to the interpolation problem through the finite Fourier Transform. (For details, see Horowitz [11]). The principle is to take the finite Fourier Transform at $n = 2^k$ points which are related to the n primitive (complex) roots of unity and then exploiting their symmetry as fully as possible. This will account for the $k = \log n$ factor in the results. Kung [20] shows both interpolation and evaluation can be solved in $0(n(\log n)^2)$operations (total), nibbling off a factor (log n) from the previous best result (Moenck and Borodin [28] through a fast division algorithm. Strassen [49] comes to the same result via deep techniques from algebraic geometry : he relates the number of MUL/DIV and ADD/SUB in a set of rational functions to the ("geometrical") degree of the intersection of an equal number of algebraic varieties, and applying a generalized version of Bezout's theorem. Moreover, he shows 0(n log n) MUL/DIV are sufficient.

Problem 2

To evaluate P_n and all its (normalized) derivatives at a point x.

We will only mention this problem because it is a beautiful example of an old and much used algorithm which suddenly appears to be not so good at all ! Classically, the solution is given as

$$T_i^{-1} = a_i + 1 \qquad i = 0, \ldots, n-1$$

$$T_j^j = a_0 \qquad j = 0, \ldots, n$$

$$T_i^j = T_{i-1}^{j-1} + x\, T_{i-1}^j \qquad j = 0, \ldots, n-1 \text{ and } i = j+1, \ldots, n$$

then

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

This is called the iterated Horner scheme, and needs $\frac{n(n+1)}{2}$ MUL and $\frac{n(n+1)}{2}$ ADD.

In 1974, Shaw and Traub [43] published the following algorithm for the same problem (see also Traub [55]) :

$$T_i^{-1} = a_{i+1}\, x^{n-i-1} \qquad i = 0, 1, \ldots, n-1$$

$$T_j^j = a_0\, x^n \qquad j = 0, \ldots, n$$

$$T_i^j = T_{i-1}^{j-1} + T_{i-1}^j \qquad j = 0, \ldots, n-1 \text{ and } i = j+1, \ldots, n$$

then

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

which needs only $3n-2$ MUL and $\frac{n(n+1)}{2}$ ADD.

In both cases the total number T(n) is of second order in n. Recently, Kung [22] has given a method requir-

ing only a total of $0(n \log n)$ operations :

$$T(n) \leqslant 6\ n \log n + 0(n \log n)$$

It is an open problem wheter there is an upper bound of order $0(n)$ for the number of additions alone. Lower bounds appear to be unknown too at this moment.

3. ON COMPUTING THE PRODUCT OF A MATRIX WITH A VECTOR OR ANOTHER MATRIX

Both polynomial evaluation and the product of a matrix with a vector can be considered as special cases of the same type of problem (Winograd [60]). Indeed they each correspond to a suitable choice for Φ, ϕ and x in the following theorem :

THEOREM (Winograd)
If we are to compute the t-vector $\psi = \Phi x + \phi$ where : Φ is a $t \times n$ matrix over the field $F(x_1, \ldots, x_n)$ of rational functions (over the field F); ϕ is a t-vector over the same field, and x is an n-vector (the "parameters" of the algorithm), then we need at least as many MUL/DIV as there are columns of Φ independent over F.
The proof, like the one for polynomials above, is again based on a substitution argument and by induction on the number of MUL/DIV operations (the substitution having for goal to remove the first "active" MUL/DIV from the computation). We remark that the x_i in the theorem are in fact the parameters of the problem, i. e. for example the coefficients of the polynomial to be computed.

COROLLARY
Put $\phi = 0$, $x =$ the pq-vector containing the elements of a $p \times q$ matrix M row by row, $\Phi = p \times pq$ matrix over the field $\mathbb{R}(y_1, \ldots, y_q)$ such that

$$\Phi x = My.$$

It follows that the ordinary method of computing My minimizes the number of multiplications, because all pq columns of Φ are independent over $\mathbb{R}$ *as functions in* $\mathbb{R}(y_1 \ \ldots \ y_q)^p$.
In the same paper, an interesting algorithm is given to compute My in about half the multiplications, using preconditioning on the rows of M :

first compute for each row

$$\xi_i = \sum_{j=1}^{n/2} x_{i,2j-1}\, x_{i,2j}$$

(Suppose M is $n \times n$ with n even)
and then we have

$$(My)_i = \sum_{i=1}^{n} x_{ij}\, y_j = \sum_{j=1}^{n/2} (x_{i,2j-1} + y_{2j})(x_{i,2j} + y_{2j-1}) - \xi_i - \eta$$

where $\eta = \sum_{j=1}^{n/2} y_{2j-1}\, y_{2j}$ needs n MUL, bringing the total to $\frac{n}{2}(n+1)$ MUL, not counting the $\frac{n}{2}$.n MUL of the ξ_i. This algorithm is easily adapted to the product of two $(n \times n)$ matrices, giving a method requiring only $[\frac{1}{2}n^3] + 0(n^2)$ MUL (all counted) instead of $n^3 + 0(n^2)$ MUL. This still is an $0(n^3)$ upper bound, and the question arises whether this can be lowered. Surprisingly, the answer is yes - thanks to an algorithm of V. Strassen [44] which is becoming a classic. It consists of an ingenuous and highly non-trivial way to multiply two 2×2 matrices in only 7 multiplications instead of 8, in a manner not using commutatively, so it can be recursively applied to larger matrices by partitioning.
This lowers the upper bound to $0(n^{\log 7})$. It works as follows :

$$\text{Let } M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad m = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$\text{Put } I = (A+D)(a+d) \quad II = (C+D)a \quad III = A(b-d)$$
$$IV = D(c-a) \quad V = (A+B)d \quad VI = (C-A)(a+b)$$
$$VII = (B-D)(c+d)$$

$$\text{Then } Mm = \begin{Bmatrix} I + IV - V + VI & III + V \\ II + IV & I + III - II + VI \end{Bmatrix}$$

The number 7 is minimal for 2×2 matrices (Winograd [62], see also Strassen [45]). Some generalizations and related results are known (Hopcroft and Kerr [10] for $2 \times n$ matrices; Gastinel [6]). Two major open problems remain :

Open Problem 1 : Can the upper bound be lowered still ?
In particular, does there exist a Strassen-like method which multiplies two 3×3 matrices in $21 (< 3^{\log 7})$ multiplications or less ? Some people suspect it to lie close to $0(n^2)$.

Open Problem 2 : Much less is known about lower bounds.
A trivial one is n^2. In fact, all known lower bounds are $0(n^2)$ (Borodin [1]). Can this be raised ? See Strassen [45] for some efforts in this direction with ortogonal matrices using the rank of the Lie ring of SO(n) (The special ortogonal group of order n).

4. A MODEL FOR ALGORITHMS

In order to prove for instance that Horner's scheme is optimal among a certain class of schemes implies the need of a good definition of "scheme" or "algorithm". Unification here seems to be lacking; most currently the definition is adapted to the problem under consideration. For polynomial evaluation often "chains" (Knuth [16])or some analogous scheme is used. This consists of computing $P_n(x)$ by a series of steps $\lambda_{-\varrho}, \ldots, \lambda_{-1}, \lambda_0, \lambda_1, \ldots, \lambda_m$ such that

1) the λ_i $(-\ell \leqslant i \leqslant -1)$ are constants

2) $\lambda_0 = x$

3) the $\lambda_j = \pm\, \lambda_k\ \omega\ \lambda_{k'}$, with $\omega \in \{+, -, *, /\}$ and $k, k' < j$

4) $\lambda_m = P_n(x)$

(also Reingold and Stocks [38]).

This principle was generalized considerably by Winograd [60] and again so by Strassen [47][48]. The latter has built a very nice formalism in which to describe elegantly a large class of problems from algebraic computational complexity. He introduces into numerical mathematics the concept of "Palgebra" (for Partial Algebra) which for a particular problem essentially is a set A of objects to be computed (the "carrier") equipped with a set Ω of operations on A (the "type") which need not be defined everywhere (hence the "partial"). With each element of ω is associated a natural number $s(\omega)$: the number of operands of ω. Thus

$$\omega : A^{s(\omega)} \nrightarrow A^{(*)}$$

Important special cases are nullary operations ($s(\omega) = 0$, constants) and unary operations ($s(\omega)=1$, for instance scalar multiplication in a vector space). Define a computation β of length l as a sequence of steps

$$\beta_i = (\omega_i, j_{i_1} \cdots j_{i_{s(\omega_i)}}) \qquad 1 \leqslant i \leqslant l,$$

with

$$\omega_i \in \Omega \text{ and } j_{i_\sigma} < i \quad \forall\, \sigma \in [1 : s(\omega_i)]^{(**)}$$

(Thus allowing no loops).

With β is associated a result sequence a_β through

$$(a_\beta)_i := a_{\beta_i} := \omega_i (a_{j_{i_1}} \cdots a_{j_{i_{s_i}}}) \quad s_i = s(\omega_i)$$

if this is defined, and else $a_\beta = \infty$. β is said to compute $F \subset A$ if 1) $\infty \notin a_\beta$

2) $F \subset a_\beta$

(Think of F for example as a set of polynomials in $A = \mathbb{R}[x]$, with x as nullary operation). Let us suppose for simplicity that all finite subsets of A are computable. (In such a case, A is called prime). After introducing an execution time $z : \Omega \to \mathbb{R}_+$, two complexity measures are associated with a computation :

$$\text{(Length) } L(z|\beta) := \sum_{i=1}^{l} z(\omega_i)$$

$$\text{(Depth) } D(z|\beta) := \max_{i=1}^{l} (z(\omega_i) + \max_{\sigma=1}^{s(\omega_i)} d_{j_{i_\sigma}})$$

The L-complexity of $F \subset A$ is defined as

$$L(z|F) = \min \{L(z|\beta) \mid a_\beta \supset F\}$$

(*) : $f : A \nrightarrow B$ stands for a partial mapping

(**) : $[m : n] = \{p \mid p \in \mathbb{N} \text{ and } m \leqslant p \leqslant n\}$

and a_β which realizes this minimum is called L-optimal.

Preconditioning can be dealt with by defining computations β which compute F mod E, i. e. which compute F if E is "known". The L and D measures have several natural properties, such as

$$L(F \cup E) \leqslant L(F \bmod E) + L(E)$$

$$D(a) \qquad \leqslant D(a \bmod E) + D(a) \quad \text{(transitivity)}$$

Two useful induction principles are also proved (L- and D-induction) in the above-mentioned papers.

5. SOME RESULTS

In his formalism, Strassen proceeds to define programs as functions from a set of inputs (palgebra's over $x_1 \ldots x_n$) to a set of so-called Ω-sets (roughly equivalent to computations). A theorem is proved which says that for almost all inputs (in a suitably generalized Zariski sense) the L-complexity of a program does not rise by restricting oneself to "straight" computations (i. e. without branching instructions). In a later paper (Strassen [50]) techniques from information theory are used to derive among other things a "statistical" result on the complexity of real numbers : for almost all $a \in \mathbb{R}$ (in the Lebesgue sense) the length $L_\epsilon(a)$ of computing an ϵ approximation b to a satisfies

$$L_\epsilon(a) \leqslant \frac{\log \frac{1}{\epsilon}}{\log \log \frac{1}{\epsilon}} + 0\left(\frac{\log \frac{1}{\epsilon}}{\log \log \frac{1}{\epsilon}}\right) \quad \text{(small)}$$

Here algebraic complexity theory seems to overlap a little with analytical theory, specifically this is closely related to some results of Paterson [35] and Kung [19]. Kung [21] also uses Strassen's formalism applied to the problem of inverting a power series :

Define L_n = the length of computation to determine the first n coefficients in $(\sum_1^\infty a_i x^i)^{-1}$

Kung proves $L_n \geqslant n + 1$

and $L_n \leqslant 4n - \log n$

by using Newton iteration, thus narrowing down the bound of Sieveking [40] ($L_n \leqslant 5n - 2$) Kung conjectures

$$L_n = 4n - 0(n).$$

Brosowski [4] suggested the generalization of Strassen's formalism to include mixed type computations (for example boolean values, thus introducing branching) and possibly loops.

6. THE COMPLEXITY OF ITERATIONS IN ONE VARIABLE : INTRODUCTION

We now turn to some problems in analytical computational complexity. Apart from the result just mentioned in § 5, the intersection with algebraic complexity theory seems to be almost empty at the

moment.
A great deal of the present efforts is directed to one particular problem, namely the approximation by means of an iteration of a zero α of a real function f : this is easy to describe and yet it has very wide applications in almost every branch of numerical mathematics. We will assume an intuitive notion of iterative method (a procedure which is repeated a number of times such that the output at step k is part of the input at step k + 1). For an early and fundamental treatment of the subject from the complexity viewpoint, we refer to Traub [53]. Some of the ideas also date back to Ostrowski [33].

Problem
Given an analytic function $f : \mathbb{R} \to \mathbb{R}$, which has a unique simple zero α in some given interval, to compute an approximation to within ϵ of α by means of an iterative method ϕ (which is supposed to be convergent).
Of course, numerous methods of solution exist for this type of problem; the difficulty is to decide which is "best". A good efficiency measure will have to satisfy some properties : it should be inversely proportional to total cost, i. e. if E_i is the efficiency of method ϕ_i applied to f, giving an ϵ-approximation to α with total arithmetic cost C_i $(i = 1, 2)$ we have

$$\frac{E_1}{E_2} \sim \frac{C_2}{C_1} \quad (A)$$

Furthermore, we should have

$$E(\phi o \phi) = E(\phi) \quad (B)$$

where $\phi o \phi$ is the iteration in which each step is a succession of two steps ϕ. In general it will also depend explicitly on the problem f. Such a measure is given by

$$E(\phi) = \frac{\log p}{d} \qquad (1)$$

Here p is the order or power of convergence of the method, defined as the greatest number $p \geqslant 1$ such that

$$\lim_{i \to \infty} \frac{|x_{i+1} - \alpha|}{|x_i - \alpha|^p} = C < \infty \quad \text{(with } C < 1 \text{ if } p = 1\text{)}$$

(It is clear that the larger p is, the greater is the speed of convergence).
d is defined as the total cost per step. Up to a monotonic function, (1) is the only possible choice (Gentleman [7]) satisfying the requirements A and B. The cost d can further be split up in (Traub [55] [56]).

1) the costs c_i of evaluating the derivatives $f^{(i)}$
2) the (minimum) cost a_ϕ of combining these evaluations to ϕ.

Thus $d = \sum_i m_i c_i + a_\phi$ with m_i the number of (new) function evaluations per step. When old evaluations are reused, the iteration is said to be with memory. Traub [54] has studied the influence of memory on the order of convergence and found that any amount of memory only adds less than 1 to the order p.

EXAMPLE. NEWTON vs. SECANT vs. TRAUB ITERATION
Newton iteration is defined as

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \equiv \phi_N(x_i, f, f')$$

In the above notation we have $a_{\phi_N} = 2$, $m_0 = m_1 = 1$, memory = 0. One can prove rather easily $p_\phi = 2$ under certain general conditions on f.
Secant iteration is given by

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

thus $a_{\phi_S} \leqslant 4$, $m_0 = 2$, $m_i = 0$ $(i \geqslant 1)$, memory = 1.

This is a special case of so-called interpolatory iteration, i. e. when x_{i+1} is computed by interpolating by polynomial of minimal degree at a number of points $x_i, x_{i-1}, \ldots, x_{i-n}$.
The order of such an iteration is given by (Traub [51]) the unique positive root p of

$$p^n - \sum_{i<n} p^i = 0$$

For ϕ_S, $p_{\phi_S} = \frac{\sqrt{5}+1}{2} \sim 1.6$

So ϕ_S has lower order than ϕ_N, but uses one new evaluation per step less. It can be shown (Traub [53]) that

$$E(\phi_N) < E(\phi_S) \text{ if } c_1 > 0.44\, c_0,$$

so Newton iteration is not always more efficient.
Both ϕ_S and ϕ_N are examples of one-point iteration (all new evaluations are at x_i).
Let us define Traub's iteration by

$$z_i = x_i + f(x_i)$$

$$x_{i+1} = \frac{(f(x_i))^2}{f(z_i) - f(x_i)} \equiv \phi_T(x_i, z_i, f)$$

This is a two-point iteration (a special case of so-called multipoint iteration) of order 2.
Recently, Kung and Traub [25] proved that for iterations without memory which use two new evaluations per step (two of f or one of f and one of f') either ϕ_N or ϕ_T is optimal relative to total operation count depending whether resp.

$$c_1 < c_0 + 3 \quad \text{or} \quad c_1 > c_0 + 3$$

Remark : In a more realistic setting, distinction should be made between multiplications count and additions count, especially where multiple precision numbers are involved.
Rissanen [39] showed that modulo a mild but complicated smoothness condition on the algorithms using one evaluation and one unit of memory, ϕ_S is optimal. This has been considerably generalized by Wozniakowski [64].

7. ITERATIONS IN ONE VARIABLE : THEOREMS AND CONJECTURES

It is clear that the order and hence the efficiency of an iterative method are in direct relation to the amount of information (new or old) used in every step. Also the way in which this information is used is of importance. We have a remarkable result of Kung [18] :

THEOREM
Let $x_{i+1} = \phi(x_i, x_{i-1}, \dots)$ be any rational iteration which uses M multiplications. Then the maximal order of ϕ is not greater than 2^M.
However (Kung [19]) this maximal order is achieved only when the limit α is rational. This generalizes a similar result of Paterson [35].
Now consider the influence of using derivatives of the function f. Winograd and Wolfe [63] proved that for any one-point iteration which at each step uses at most evaluations of f and its first d derivatives, the maximal order is $d+2$.
This is done by establishing the existence of an analytic f such that

$$\liminf_{k\to\infty} \frac{|\phi_k(x_k, \dots) - \alpha|}{\prod_{j\leqslant k} |x_j - \alpha|^{d+1}} \quad \text{exists and is} > 0.$$

Remark that the iteration method need not be stationary (ϕ_k) and any amount of memory is permitted. Without memory, the maximal order is $d+1$; iterative methods exist which realize this bound (see Kung and Traub [24], who show indeed that any one-point iteration without memory of order $d+1$ must use evaluations of f and its first d derivatives).
This method is based on interpolation; in fact, Wozniakowski [64] shows (Hermite-) interpolatory methods are optimal in a very general sense. The above restrictions do not apply to multipoint iterations, at is shown by examples of King [13], exhibiting fourth order three-point methods using two f', one f, and one f', two f evaluations respectively.
Kung and Traub [23] then found two families of n-point iterations, both of order 2^{n-1}, using either n evaluations of f (and no derivatives) or $n-1$ evaluations of f and only 1 of f'. The former is given by

$$z_{i,1} = x_i + \beta f(x_i) \qquad (\beta : \text{real constant})$$

$$z_{i,k} = \underline{p}_k^{-1}(0)$$

$$x_{i+1} = z_{i,n} = \underline{p}_n^{-1}(0)$$

where P_k^{-1} is the inverse interpolating polynomial at $f(z_{i,j})$ $1 \leqslant j \leqslant k-1$.

This is about the fastest stationary iteration method known at this moment.
They also show :

THEOREM
If one uses (Hermite) interpolation, 2^{n-1} is the maximal order obtainable with n evaluations (for methods such as above).
Also, for any rational method, the order cannot be more than 2^n.
They conjecture however, that 2^{n-1} is optimal. Recently, they proved this in the case $n = 2$ (Kung and Traub [25]); $n \geqslant 3$ remains open, but in any case, absolute bounds on computational efficiency are already in view.

8. A FEW OTHER AREAS OF RESEARCH IN ANALYTIC COMPLEXITY

First of all, we have of course the multi-dimensional iteration calculus. Very little is known here, because the difficulties are vast : the notion of "order" cannot indeed be generalized to n dimensions without at least some care (Ortega and Rheinboldt [32])
Brent [2] [3] defines the order as

$$\rho = \lim_{i\to\infty} \frac{\log \| |x_{i+1} - \alpha| \|}{\log \| |x_i - \alpha| \|}$$

(where x_i, x_{i+1} and α are n-dimensional vectors).
Wozniakowski [64] defines the order via test-sequences (i. e. by comparing the speed of convergence of ϕ with sequences of given order ρ). He remarks the difficulty lies in the fact that in the multivariate case, one cannot in general be sure that the iteration function remains defined on the previously computed results. He proposes the investigation of assumptions of "good position" in results concerning convergence. With his definition, he is able to prove the optimality of interpolatory methods in the one-point case (using derivatives). Really good efficiency measures are yet to be found;
Brent [2] defines it as $\frac{\log \rho}{\text{cost per step}}$ and conjectures the maximal efficiency is bounded by $\log \rho_{\infty,n}$ where $\rho_{\infty,n}$ is the (unique) positive root of

$$\sum_j \rho^{-\binom{n+j}{j}} = 1$$

when using only evaluations of f, and taking one evaluation of the vector f as unit of cost.
In many cases, the cost of evaluating derivatives is prohibitive, and the cost of memory, which may seem trivial in the one-dimensional case, can become

important. Brent [3] discusses also some classes of practical, yet efficient methods. These are all based on some relatively simple iteration procedure, such as discrete Newton, secant, or triangularization, which is repeated at each "step" a number of times (this number is then optimized and depends in general on the number of equations in the problem under consideration).

Quite another domain of interest is the complexity of finite grid methods for the solution of partial differential equations. Even less is known here. We have a result of Schultz [42] who develops a method with so-called cubic Hermite functions on a "square root" grid needing $0(N^2)$ arithmetic operations and $0(N^2)$ storage when $N \to \infty$; this is asymptotically "optimal" in view of the fact that N^2 are always needed, even if we know the solution and have to tabulate it. Completely different techniques are employed by Hoffman, Martin and Rose [9] who prove $0(n^3)$ multiplications and $0(n^2 \log n)$ storage is necessary when one uses elimination procedures. They depend on graph-theoretical techniques by representing a matrix with its rows as vertices and its non zero elements as edges of a graph; elimination is then done by deleting a vertex and interconnecting all its neighbours.

9. CONCLUSION

Computational complexity of practical problems constitutes a fast-growing field of interest and research. A better insight in the theory of the computing process is being obtained. Many areas however remain untouched as yet or almost so : Miller [26] has started to develop a theory of "infinitely" complex problems, making use of topology and analysis; complexity theory for numerical integration or for finite difference methods in the numerical solution of ordinary differential equations is virtually nonexistent at this moment. Also, the coming of the new parallel or vector machines will require the development of quite new techniques and algorithms (Traub [57], Winograd [61]). An important practical problem associated with the ever "better" algorithms is their numerical stability, i. e. their sensitivity to rounding errors etc... Except for some isolated efforts, these problems are open (W. Miller [27], Ortega [31], Tienari [52], Wilkinson [59]).

(Take for example Traub's iteration in § 6 which involves the division of very small numbers when convergence proceeds). Most of the problems seem to originate from the difficult marriage between mathematical analysis and the finiteness of the machine, so good models will have to be developed, especially for analytical complexity theory (Tsichritzis [58]).

BIBLIOGRAPHY AND REFERENCES

The following abbreviations are used :

Miller & Thatcher : = Complexity of Computer Computations, Miller & Thatcher (Ed.) Plenum Press 1972.

Report CMU : = Report Dept. Comp. Sc. Carnegie Mellon Univ. Pittsburg, Penn.

1. Borodin A., "Computational complexity : Theory and practice" from : AHO (ed.) 1972
2. Brent R., "The computational complexity of iterative methods for systems of equations". Miller & Thatcher
3. Brent R., "Some efficient algorithms for solving systems of nonlinear equations". Siam J. Num. Anal. Vol. 10,no 2 April 1972
4. Brosowski, "Private communication"
5. Byrne & Hall (Ed.), "Numerical solution of systems of nonlinear equations". Acad. Press 1973
6. Gastinel, "Sur le calcul des produits de matrices". Num. Math. Vol. 17 (1971)
7. Gentleman W. M., "On the relevance of various cost models of complexity. Complexity of sequential and parallel algorithms", JF Traub (Ed.) Acad. Press 1973
8. Hartmanis J. and Hopcroft J. E., "An overview of the theory of computational complexity". J. ACM Vol. 18, no 3, July 1971
9. Hoffman A. J., Martin S. M. and Rose D. J., "Complexity bounds for regular finite difference and finite elements grids". SIAM J. Num. Anal. Vol. 10, no 2, April 1973
10. Hopcroft J. E. and Kerr L. R., "On minimizing the number of multiplications necessary for matrix multiplication". SIAM J. Appl. Math. Vol. 20, no 1, Jan. 1971
11. Horowitz E., "A fast method for interpolation using preconditioning". Information processing letters 1 (1972)
12. Isaacson E. and Keller H. B., "Analysis of numerical methods". John Wiley 1966
13. King R. F., "A family of fourth order methods for nonlinear equations". SIAM J. Numerical Anal. Vol. 10, no 5 (Oct. 1973)
14. Knuth, D. E., "Mathematical analysis of algorithms". IFIP 1971 Congress Proceedings I135-I143
15. Knuth, D. E., "The art of computer programming : Vol. 1" (Fundamental algorithms) Addison Wesley 1968
16. Knuth, D. E., "The art of computer programming : Vol. 2" (Seminumerical algorithms) Addison Wesley 1971
17. Knuth, D. E., "The art of computer programming : Vol. 3" (Sorting and searching) Addison Wesley 1973
18. Kung, H. T., "A bound on the multiplication efficiency of iteration". J. Comp. & Syst. Sciences Vol. 7 no 4 (1973)
19. Kung, H. T., "The computational complexity of algebraic numbers". Report CMU (1973)
20. Kung, H. T., "Fast evaluation and interpolation". Report CMU (1973)
21. Kung, H. T., "On computing reciprocals of power series". Report CMU (1973)
22. Kung, H. T., "A new upper bound on the complexity of derivative evaluation". Report CMU (1973)
23. Kung, H. T. and Traub, J. F., "Computational complexity of one-point and multipoint iteration". Report CMU (1973)
24. Kung, H. T. and Traub, J. F., "Optimal order of one point and multipoint iteration". Report CMU (1973)
25. Kung, H. T. and Traub, J. F., "Optimal order and efficiency for iterations with two evaluations". Report CMU (1973)
26. Miller, W., "Toward abstract numerical analysis". J. ACM Vol. 20, no 3, July 1973

27. Miller, W., "Remarks on the complexity of roundoff analysis". IBM Report (1973)

28. Moenck, R. and Borodin, A., "Fast modular transform via division". Proc. 13th Symp. on switching and automata theory 1972

29. Motzkin, T. S., "Bull. Amer. Math. Soc." Vol. 61 (1955)

30. Munro, I. and Paterson, M. S., "Optimal algorithms for parallel polynomial evaluation". J. Comp. & Syst. Sciences, Vol. 7, no 2 (1973)

31. Ortega, J. M., "Stability of difference equations and convergence of iterative processes". SIAM J. Num. Anal. Vol. 10, no 2, April 1973

32. Ortega, J. M. and Rheinboldt, W. C., "Iterative solution of nonlinear equations in several variables". Academic Press 1970

33. Ostrowski, A. M., "Solution of equations and systems of equations". Academic Press 1960

34. Pan, V. Y., "Methods of computing values of polynomials". Russian Math. Surveys, Vol. 21 (1966)

35. Paterson, M. S., "Efficient iteration for algebraic numbers". Miller & Thatcher (1972)

36. Rall, L. B., "Computational solution of nonlinear operator equations". John Wiley 1969

37. Rabin, M. O. and Winograd, S., "Fast evaluation of polynomials by rational preparation". Comm. Pure and Appl. Math. Vol. 25 (1972)

38. Reingold, E. M. and Stocks, A. I., "Simple proofs of lower bounds for polynomial evaluation". Miller & Thatcher (1972)

39. Rissanen, J., "On optimum root-finding algorithms". J. Math. Anal. Applic. Vol. 36, no 1, Oct. 1971

40. Sieveking, M., "An algorithm for division of powerseries". Computing Vol. 10 (1972)

41. Schönhage, K. und Strassen, V., "Schnelle Multiplikation Grosser Zählen". Computing Vol. 7 (1971)

42. Schultz, M. H., "The computational complexity of elliptic partial differential equations". Miller and Thatcher (1972)

43. Shaw, M. and Traub, J. F., "On the number of multiplications for the evaluation of a polynomial and some of its derivatives". J. A.C.M. Vol. 21, no 1 (1974)

44. Strassen, V., "Gaussian elimination is not optimal". Num. Math. Vol. 13, 1969

45. Strassen, V., "Vermeidung von Divisionen". Journal für Mathematik, Band 264 (1972)

46. Strassen, V., "Evaluation of rational functions". Miller & Thatcher (1972)

47. Strassen, V., "Berechnung und Programm I". Acta Informatica, Vol. 1, 1972

48. Strassen, V., "Berechnung und Programm II". Acta Informatica, Vol. 2, 1973

49. Strassen, V., "Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten". Num. Math. Vol. 20 (1973)

50. Strassen, V., "Berechnungen in Algebren endlichen Typs", Computing Vol. 11, 1973

51. Strassen, V., "Polynomials with rational coeff. which are hard to compute". Report Universität Zürich 1973

52. Tienari, M., "Some topological properties of numerical algorithms", BIT Vol. 12, 1972

53. Traub, J. F., "Iterative methods for the solution of equations". Prentice Hall (1964)

54. Traub, J. F., "Computational complexity of iterative processes". SIAM J. Comput. Vol. 1, no 2, June 1972

55. Traub, J. F., "Theory of optimal algorithms". Report CMU, 1973

56. Traub, J. F., "An introduction to some current research in numerical computational complexity". Report CMU, 1973

57. Traub, J. F., "Iterative solution of tridiagonal systems in parallel or vector computers". Report CMU, 1973

58. Tsichritzis, D., "A model for iterative computation". Information Sciences, Vol. 5, 1973

59. Wilkinson, "Rounding errors in algebraic processes". Prentice-Hall N. Y. (1963)

60. Winograd, S., "On the number of multiplications necessary to compute certain functions". Comm. Pure and Appl. Math., Vol. 23, 1970

61. Winograd, S., "Parallel iteration methods". Miller & Thatcher 1972

62. Winograd, S., "On multiplication of 2 x 2 matrices". IBM Report RC2767

63. Winograd, S. and Wolfe, P., "Optimal iterative processes" IBM Research Report (Aug. 1971)

64. Wozniakowski, H., "Maximal stationary iterative methods for the solution of operator equations". Report CMU 1973