71-15,747

COHEN, Arthur Ira, 1945-
RATE OF CONVERGENCE AND OPTIMALITY PROPERTIES
OF ROOT FINDING AND OPTIMIZATION ALGORITHMS.

University of California, Berkeley, Ph.D.,
1970
Engineering, electrical

University Microfilms, A XEROX Company, Ann Arbor, Michigan

Rate of Convergence and Optimality Properties of Root Finding and Optimization Algorithms

By

Arthur Ira Cohen

B.S. (Cornell University) 1965
M.S. (Cornell University) 1967

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Approved:

[illegible]

[illegible]

Richard M. Karp

Committee in Charge

DEGREE CONFERRED SEPT. 18, 1970

Acknowledgements

I wish to thank Professor Pravin Varaiya for his excellent guidance and advice throughout this study. I would also like to sincerely thank my wife for her help and encouragement.

I also acknowledge the National Aeronautics and Space Administration for its support.

Contents

Abstract

A class of optimal root finding algorithms are described which include the usual interpolatory methods such as the Newton-Raphson and secant algorithms. Rate of convergence is found for several specific algorithms. In particular a multipoint secant method is developed which uses no derivatives and yet converges to the root of systems of equations with the same rate as the Newton-Raphson method. Also rate of convergence is found for several conjugate gradient algorithms when they are used to minimize nonlinear, nonquadratic functions on R^n.

Root finding algorithms can be classified by the amount of information or data used at each step. If one assumes infinite precision, it is possible to encode past information (memory) into a data point. We have added a condition to the definition of rate which insures that encoding does not improve the rate. Using this new definition, the class of optimal algorithms is obtained.

The rate of convergence of the multipoint secant method is found with the help of an interpolatory error formula that is developed. The formula compares a function mapping R^n into R^n with an affine interpolation which agrees with the function at n + 1 points.

Rate of convergence of the conjugate gradient algorithms is found by comparing them with the Newton-Raphson method. It is shown that n steps of these algorithms approach the minimum with rate two. This result is also shown to hold for the case where the conjugate variable is reinitialized every r steps (where r is greater or equal n).

I. Introduction

The classification, convergence and rate of convergence of root finding and unconstrained optimization algorithms have been studied extensively by Ostrowski[8], Traub[11], and others[6],[7]. These algorithms have been classified by the amount of new and old data (the old data used is called memory) used at each step. Rate of convergence has been defined and has been calculated for many algorithms. It has been assumed in these papers (and it shall be assumed here) that one knows exactly all the data, that is, there are no truncation or other errors. Winograd[†] has pointed out that this can lead to ambiguity in the notion of memory since instead of using old information explicitly at each step, one can use it implicitly by embedding it in other data. Since it is desired to find the best algorithms belonging to a certain class, this ambiguity must be dealt with. One can do this in two ways. The first is by putting strict conditions on the types of algorithms you will allow [11] and the second is by changing the definition of rate so that it will not be affected by implicit information. The second method is used in this paper.

After defining classes of iteration functions and rate of convergence, we will show that certain algorithms are best in a particular class. It turns out that the best or optimal algorithms are not unique, however it will be shown that the standard interpolatory algorithms are among the optimal ones.

[†] Lecture at the University of California, Berkeley during the Fall of 1968.

We will then be concerned with giving rates for specific algorithms. In particular an algorithm, called the multipoint secant method, has been developed which uses no derivatives and yet converges to the root of functions from R^n into R^n with the same rate as the Newton-Raphson algorithm.

Finally, we find the rate of convergence of several conjugate gradient algorithms when they are used to minimize a class of nonlinear, nonquadratic, real valued functions on R^n. Conjugate gradient algorithms use a one-dimensional minimization at each step which, if done exactly, would in general require an infinite number of function evaluations. For this reason we cannot compare it with methods that use a finite amount of data at each step. This section of the thesis is therefore a slight digression from the rest in that it does not attempt to give any optimality properties for these algorithms.

II.1 Notation

The notation used can best be described with the help of an example. Consider the secant method when it is used to find the roots of functions $f \in \mathcal{F}$ mapping $R \to R$. It is

$$x_{k+1} = x_k - \left(\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right) f(x_k) \quad .$$

We define the data at the kth step by $d(f,\underline{x}_k) = (x_k, x_{k-1}, f(x_k), f(x_{k-1}))$, where $\underline{x}_k = (x_k, x_{k-1})$, and let $\mathcal{D}(\mathcal{F})$ be the set of all data that can be obtained from $\mathcal{F}$. We are interested in the asymptotic behavior of the algorithm therefore we assume that the algorithm does not reach the root in a finite number of steps. Thus x_k does not equal x_{k-1} for all k since their being equal would imply that x_k was a root of f.

The algorithm can now be considered as a map Φ from $\mathcal{D}(\mathcal{F})$ into R, that is

$$x_{k+1} = \Phi(d(f,\underline{x}_k)) = \Phi(x_k, x_{k-1}, f(x_k), f(x_{k-1})) \quad .$$

Since the value of f at x_{k-1} needed at the kth step has been already calculated at the (k-1) step, we say that this algorithm has memory.

In general the following notation is used:

(i) x, x_k or x_k^j are points in R^n.

(ii) $\mathcal{F}$ is the set containing the functions whose roots (or extremum) we want to find.

(iii) $\underline{x}_k = (x_k^0, x_k^1, \ldots, x_k^s)$ is an $(s+1)$-tuple called the data point at the kth step. It consists of those points, at the kth step, for which it is necessary to know f. We shall assume all the elements of $\underline{x}_k$ are different.

(iv) $d(f,\underline{x}_k)$ is the data used at the kth step. We sometimes write $d(f,\underline{x}_k) = (d^0(f,\underline{x}_k), \bar{d}(f,\underline{x}_k))$ where $d^0(f,\underline{x}_k) = \underline{x}_k$ and $d(f,\underline{x}_k)$ is the rest of the data.

(v) $\mathcal{D}(\mathcal{F})$ is the set of all data, $d(f,\underline{x}_k)$, that can be obtained from $\mathcal{F}$.

Also:

(vi) $f'(x) = f^{(1)}(x) = Df(x) = D^1 f(x)$ is the derivative of f at x. $D_j f(x)$ is the partial derivative of f with respect to the jth component of x.

(vii) $f^{(j)}(x) = D^j f(x)$ is the jth derivative of f at x. If $f: R^n \to R^m, n>1$, we shall consider $D^j f(x)$ to be a multilinear map from $R^n \times \ldots \times R^n$ (j times) into R^m defined by

$$D^j f(x)(t_1,\ldots,t_j) = \sum_{i_1=1}^{n} \cdots \sum_{i_j=1}^{n} D_{i_1} \cdots D_{i_j} f(x) t_{1,i_1} \cdots t_{j,i_j}$$

where $t_k = (t_{k,1}, t_{k,2}, \ldots, t_{k,n})$. We also define

$$D^j f(x)\, t^j = D^j f(x)\underbrace{(t,\ldots,t)}_{j \text{ elements}} .$$

(See Dieudonné [3] for justification of this notation). We shall call $D^j f(x)$ nonsingular if $D^j f(x)(t_1,\ldots,t_j) = 0$ implies $t_k = 0$ for some $k\varepsilon\{1,2,\ldots,j\}$.

(viii) $|x|$ denotes the absolute value of x if $x\varepsilon R$ and the norm of x when $x\varepsilon R^n$. If the type of norm makes a difference in a particular theorem then it shall be specified at that time.

(ix) $B_r(\alpha) = \{x : |x-\alpha|<r\}$.

(x) $co(x_1,\ldots,x_n) = \{y = \sum_{i=1}^{n} \lambda_i x_i : \lambda_i \geq 0 \text{ for } i = 1,2,\ldots,n, \sum_{i=1}^{n} \lambda_i = 1\}$

(xi) $co(\underline{x}_k, z_1, z_2, \ldots, z_n) = co(x_k^0,\ldots,x_k^s, z_1,\ldots,z_n)$.

II.2 Classification of Algorithms

The types of data we will consider fall into four main classes.

(i) One point: The data is made up of f and its derivatives evaluated at one point x_k, (e.g. $d(f,x_k) = (x_k, f(x_k), f^{(1)}(x_k),\ldots,f^{(q-1)}(x_k))$).

(ii) One point with memory: The data is made up of f and its derivatives evaluated at one point x_k and of reused information from past times, (e.g. $d(f,\underline{x}_k) = (x_k, x_{k-1},\ldots,x_{k-s}, f(x_k), f^{(1)}(x_k), \ldots, f^{(\gamma_0-1)}(x_k), f(x_{k-1}),\ldots,f^{(\gamma_1-1)}(x_k),\ldots,f(x_{k-s}), \ldots, f^{(\gamma_s-1)}(x_{k-s})$)).

(iii) Multipoint: The data is made up of f and its derivatives evaluated at x_k and at other points $z_1(f,x_k),\ldots,z_j(f,x_k)$. The $z_i(f,x_k)$ $i = 1,\ldots,j$ are given functions of f and its derivatives evaluated at x_k.

(iv) Multipoint with memory: The data is made up of evaluations of f and its derivatives at x_k, at other points $z_1(f,x_k),\ldots,z_j(f,x_k)$, and of reused information from past times.

A schematic diagram of an algorithm is given in Figure 1.

II.3 Definition of Algorithm

The properties of an algorithm are dependent on the class of functions $\mathcal{F}$ on which they act. We shall, unless stated otherwise, assume that $\mathcal{F} \subset C^\infty(R^n)$, the space of infinitely differentiable functions. This assumption is made to make the discussion more readable; it shall be clear from the context when it can be relaxed. We shall now define what we mead by an algorithm.

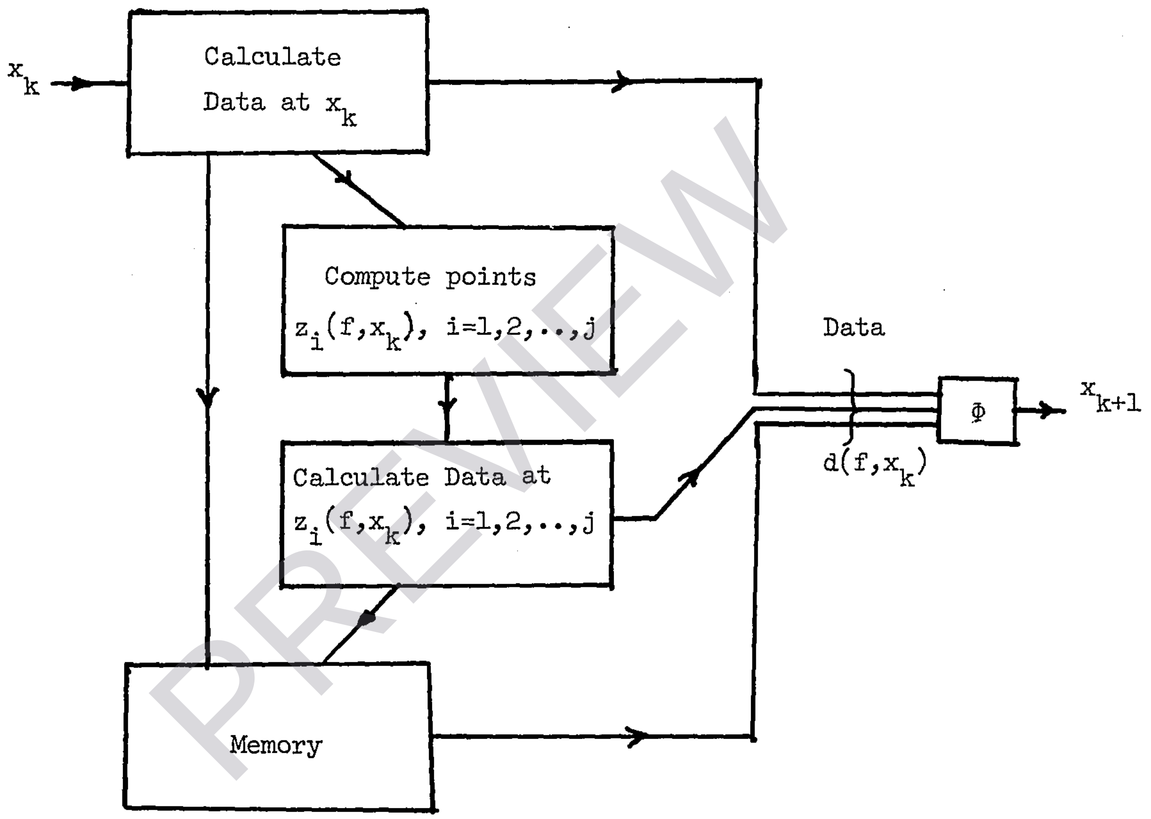
(1) Definition: $\Phi:\mathcal{D}(\mathcal{F})\to R^n$ is an algorithm on $\mathcal{D}(\mathcal{F})$, denoted $\Phi \in \mathcal{A}(\mathcal{D}(\mathcal{F}))$, if for all f in $\mathcal{F}$ there exists a neighborhood T_f of one of the roots of f, $\alpha(f)$, such that for all initial conditions depending on data on T_f, the sequence $x_1,x_2,\ldots$ formed by $x_{i+1} = \Phi(d(f,\underline{x}_i))$ satisfies

(2) (a) $x_i \in T_f$ for $i = 1,2,\ldots$

(3) (b) $x_i\to\alpha(f)$ as $i\to\infty$.

We shall call such a sequence a solution sequence for f.

Remark: The initial data, $d(f,\underline{x}_0)$, is made up of f and its derivatives evaluated at arbitrary distinct points in T_f. So if, for example, $d(f,\underline{x}_k) = (x_k,x_{k-1},z(f,x_k);f(x_k),f(x_{k-1}),f(z(f,x_k)))$ then $d(f,\underline{x}_0) = (x^0,x^1,z(f,x^0),f(x^0),f(x^1),f(z(f,x^0)))$ where x^0 and x^1 are arbitrary distinct elements of T_f.



x_k
Calculate Data at x_k
Compute points $z_i(f,x_k)$, i=1,2,..,j
Calculate Data at $z_i(f,x_k)$, i=1,2,..,j
Memory
Data
$d(f,x_k)$
Φ
x_{k+1}

Figure 1. A Schematic Diagram of an Algorithm

II.4 Implicit Use of Past Data and Rate of Convergence

We can implicitly use at step k past information, not given explicitly in $d(f,\underline{x}_k)$, by encoding it into the data point. For example, let $f:R\to R$ and let Φ be an algorithm on a data space $\mathcal{D}$ containing elements of the form $d(f,\underline{x}_k) = (x_k,x_{k-1},x_{k-2},f(x_k),f(x_{k-1}),f(x_{k-2}))$. We shall construct a new algorithm $\tilde{\Phi}$ which acts essentially the same as Φ but is defined on a data space $\tilde{\mathcal{D}}$ containing elements of the form $\tilde{d}(f,\underline{x}_k) = (x_k,x_{k-1},f(x_k),f(x_{k-1}))$. This shall be done with help of a map h from $R\times R\times R_u\times R_u\to R$ which we shall denote by $h(a,b,c,d) = [a.b.c.d]$ (where $R_u = \{x \in R : x<1\})$. H encodes the three points a,b,c, into one point. To define h exactly we must express a,b,c, and d in decimal notation:

$$a = \operatorname{sgn}(a)\ a_p a_{p-1}\cdots a_0.a_{-1}a_{-2}\cdots$$
$$b = \operatorname{sgn}(b)\ b_q b_{q-1}\cdots b_0.b_{-1}b_{-2}\cdots$$
$$c = \operatorname{sgn}(c)\ 0.0\ldots 0c_{-r}c_{-r-1}\cdots$$
$$d = \operatorname{sgn}(d)\ 0.0\ldots 0d_{-s}d_{-s-1}\cdots$$

Then define

$$[a.b.c.d] = \operatorname{sgn}(a)a_p a_{p-1}\cdots a_0.a_{-1}\cdots a_{-ms}b_{q+1}c_{r+1}a_{-ms-1}b_q c_r\cdots$$

where b_{q+1} (or c_{r+1}) is 1 if b (or c) is positive and is 0 if b(or c) is negative, and where m is specified below. Note that if we are given [a.b.c.d] and d, we can exactly determine a,b, and c. We now define

$$\tilde{\Phi}(x_k,x_{k-1},f(x_k),f(x_{k-1})) = [\Phi(a_k,x_{k-1},b_k,f(x_k),f(x_{k-1}),c_k).x_{k-1}.f(x_{k-1}).f(x_k)]$$

where $x_k = [a_k.b_k.c_k.d_k]$. So if (x_2,x_1,x_0) are the initial points used for Φ , producing the sequence $\{z_k\}$, and if $([x_2.x_0.f(x_0).f(x_1)],x_1)$ is used for $\tilde{\Phi}$, producing the sequence $\{x_k\}= \{[a_k.b_k.c_k.d_k]\}$, then $z_k = a_k$ for all k and $|z_k - x_k| \leq 10^{-ms}$. So if $z_k \to \alpha$ with power p (power is defined later in this section) we can, since $f(x_k) \to 0$, make m large enough to insure that $x_k \to \alpha$ with power p also.

Clearly $\tilde{\Phi}$ cannot be compared to an algorithm such as the secant method which does not use encoding. Therefore, in order to answer "What is the best algorithm given a particular amount of data?", we must eliminate from consideration algorithms that encode. The most direct way to accomplish this is to just consider algorithms that do not encode; however, it is not clear how to mathematically describe this class. We shall instead add a condition to the definition of rate which insures that encoding does not increase the rate.

The rate or power of an algorithm $\Phi \in \mathcal{A}(\mathcal{D}(\mathcal{F}))$ is defined by Ostrowski [8] as the largest constant p>1 such that for all solution sequences $\{x_k\}$ and for all $f \varepsilon \mathcal{F}$ there exists a constant C so that

$$\overline{\lim_{k\to\infty}} \frac{|\Phi(d(f,x_k)) - \alpha(f)|}{|x_k - \alpha(f)|^p} \leq C < \infty \ .$$

The condition we shall add requires that the error $|x_k - \alpha(f)|$ is decreased by the same order for a large class of functions whose data at step k is the same (see part (iii) in the definition below). This will insure that information not given explicitly in d will not increase the rate. It should be emphasized that encoding does indeed increase the speed at which an algorithm can converge and so there is nothing wrong with the original definition; it prevents us, however, from comparing algorithms meaningfully.

The new definition is:

(4) Definition: $\Phi \varepsilon \mathcal{A}(\mathcal{D}(\mathcal{F}))$ has uniform rate or uniform power p if:

(i) for all $f \varepsilon \mathcal{F}$, there exists a constant C_1 such that

$$(5) \qquad \overline{\lim_{k\to\infty}} \frac{|\Phi(d(f,x_k)) - \alpha(f)|}{|x_k - \alpha(f)|^p} \leq C_1 < \infty$$

where $\{x_k\}$ is any solution sequence on f,

(ii) there exists an $h \varepsilon \mathcal{F}$ and a solution sequence, $\{z_k\}$, on h such that

(6) $$\underline{\lim_{k\to\infty}} \frac{|\Phi(d(h,\underline{z}_k)) - \alpha(h)|}{|z_k - \alpha(h)|^p} > 0 ,$$

(iii) for all $h \in \mathcal{F}$ and solution sequences $\{z_k\}$ on h for which (6) occurs and for all sequences of functions $\{g_k\}$ in $\mathcal{F}$ satisfying

(7) (a) $d(g_k,\underline{z}_k) = d(h,\underline{z}_k)$

(8) (b) $g_k \to h$ uniformly at $\alpha(h)$, that is, for all $\varepsilon>0$ there exists $\delta>0$ and K such that $k \geq K$ and $z \in B_\delta(\alpha(h))$ implies

$$|g_k(z) - h(z)| + |Dg_k(z) - Dh(z)| < \varepsilon$$

(c) $\{D^j g_k(\xi_k)\}$ is a Cauchy sequence for all $j = 1,2,\ldots$ and sequences $\{\xi_k\}$ where $\xi_k \in co(\underline{z}_k,\alpha(h))$,

there exists a constant $C_2(h)$ and roots $\alpha(g_k)$ of g_k such that

(9) $$\overline{\lim_{k\to\infty}} \frac{|\Phi(d(h,\underline{z}_k)) - \alpha(g_k)|}{|z_k - \alpha(g_k)|^p} \leq C_2 < \infty .$$

(10) Remark: Condition (c) is required to insure that the new definition of rate does not break down when we are analyzing algorithms with memory. If we were just considering memoryless algorithms it would not be required.

(11) Remark: If we assume that $Dh(\alpha(h))$ is invertible, condition (b)

implies that g_k has a root $\alpha(g_k)$ such that $|z_k - \alpha(g_k)|$ goes to zero with the same rate as $|z_k - \alpha(h)|$. To be precise:

(12) Lemma: Suppose that for all functions $f \in \mathcal{F}$, $f \in C^1(R^n)$, and $Df(\alpha(f))$ is invertible. Let $\{x_k\}$ be any sequence converging to $\alpha(f)$ and let $\{g_k\}$ be a sequence of functions such that $g_k(x_k) = f(x_k)$ and $g_k \to f$ uniformly at $\alpha(f)$, then there exists m, M, K and a root $\alpha(g_k)$ of g_k such that $k \geq K$ implies

$$0 < m \leq \frac{|x_k - \alpha(f)|}{|x_k - \alpha(g_k)|} \leq M < \infty \quad .$$

The proof of this Lemma is found in Appendix A. This result is used in many of our proofs, therefore we shall usually assume that $Df(\alpha(f))$ is invertible.

II.5 Sufficient Algorithms

We shall now construct optimal algorithms. An algorithm in $\mathcal{A}(\mathcal{D}(\mathcal{F}))$ will be called optimal if its power is greater or equal the power of any other algorithm in $\mathcal{A}(\mathcal{D}(\mathcal{F}))$. We will begin by looking at a particular subset of $\mathcal{F}$.

(13)Definition: $\hat{\mathcal{F}} \subset \mathcal{F}$ is a sufficient class of functions for $\mathcal{D}(\mathcal{F})$ if for all $\omega \in \mathcal{D}(\mathcal{F})$, there exists a unique $\hat{f}_\omega \in \hat{\mathcal{F}}$ such that $d(\hat{f}_\omega, \omega^0) = \omega$. In addition we require that:

(i) $\hat{f}_{d(f,\underline{x}_i)}$ converges pointwise to the same function for all sequences of data points $\underline{x}_i \to \underline{x}$. If $\underline{x}$ is a data point then $\hat{f}_{d(f,\underline{x}_i)} \to \hat{f}_{d(f,\underline{x})}$. If $\underline{x}$ is not a data point then we define $\hat{f}_{d(f,\underline{x})} = \lim_{i\to\infty} \hat{f}_{d(f,\underline{x}_i)}$.

(ii) For all $\varepsilon>0$ there exists a $\delta>0$ such that

$$|\hat{f}_{d(f,\underline{x})}(x) - f(x)| + |D\hat{f}_{d(f,\underline{x})}(x) - Df(x)| < \varepsilon$$

for all x and $\underline{x}$ such that $co(x,\underline{x}) \subset B_\delta(\alpha(f))$.

(iii) $D^j\hat{f}_{d(f,\underline{x})}(x)$ is continuous in $(\underline{x},x)$ at $(\underline{\alpha(f)},\alpha(f))$ for $j = 1,2,\ldots$, where $\underline{\alpha(f)} = (\alpha(f),\ldots,\alpha(f))$.

(14) Remark: If $\{x_k\}$ is a solution sequence for f, then $\{\hat{f}_{d(f,\underline{x}_k)}\}$ satisfies conditions (a), (b), and (c) required of $\{g_k\}$ in definition of rate.

If we consider $\hat{f}_{d(f,\underline{x}_k)}$ as an approximation to f, then we arrive naturally at the following method of obtaining the root of f.

(15) Definition: $\hat{\Phi} \in \mathcal{A}(\mathcal{D}(\mathcal{F}))$ is a sufficient algorithm if there exists a sufficient class of functions $\hat{\mathcal{F}}$ such that $\hat{\Phi}(\omega) = \alpha(\hat{f}_\omega)$ for all $\omega \in \mathcal{D}(\mathcal{F})$, where $\alpha(\hat{f}_\omega)$ is some root of $\hat{f}_\omega$.

(16) Example: If $\mathcal{D}(\mathcal{F})$ contains data of the form $\omega_0 = (x_0,f(x_0),Df(x_0))$ where $Df(x_0)$ is assumed invert ble, then we can define the function $\hat{f}_{\omega_0}(x) = f(x_0) + Df(x_0)(x-x_0)$. $\hat{\mathcal{F}} = \{\hat{f}_\omega : \omega \in \mathcal{D}(\mathcal{F})\}$ will be a sufficient class of functions and $\hat{\Phi}(\omega_0) = x_0 - (Df(x_0))^{-1}f(x_0)$ will be a sufficient algorithm. $\hat{\Phi}$ is the Newton-Raphson algorithm.

II.6 Optimal Sufficient Algorithms

Under certain conditons sufficient algorithms are optimal. To show this, it is first necessary to give the relationship between the data $\underline{x}_k = (x_k^0,\ldots,x_k^n)$ and x_{k+1} for a sufficient algorithm used to find roots of functions from $R \to R$.

(17) Lemma: Let $\mathcal{F} \subset C^\infty(R)$, let $d(f,\underline{x}_k) = (x_k^0,\ldots,x_k^n, f(x_k^0),\ldots,f^{(\gamma_0-1)}(x_k^0), f(x_k^1),\ldots,f^{(\gamma_n-1)}(x_k^n))$, and let $\hat{\mathcal{F}}$ be a sufficient class of functions for $\mathcal{D}(\mathcal{F})$. Let $\hat{\Phi}$ be defined on $\hat{\mathcal{F}}$, and let $\hat{f}_k \equiv \hat{f}_{d(f,\underline{x}_k)}$. Then there exists, for each $t_0 \in R$, a $\xi \in co(\underline{x}_k, t_0)$ such that

$$(18) \qquad f(t_0) = \hat{f}_k(t_0) + \frac{1}{q!}\left[f^{(q)}(\xi) - \hat{f}_k^{(q)}(\xi)\right] \quad w(t_0)$$

where $q = \sum_{j=0}^{n} \gamma_j$ and $w(t) = \prod_{j=0}^{n} (t-x_k^j)^{\gamma_j}$. Also if $\hat{f}_k'(x) \neq 0$ on $co(x_{k+1},\alpha)$, then there exists an $\eta_{k+1} \in co(x_{k+1},\alpha)$ and $\xi_k \in co(\underline{x}_k,\alpha)$ such that

$$(19) \qquad |x_{k+1} - \alpha| = H_{k+1} \prod_{j=0}^{n} |x_k^j - \alpha|^{\gamma_j}$$

where $H_{k+1} = \dfrac{1}{q!}\left|\dfrac{f^{(q)}(\xi_k) - \hat{f}_k^{(q)}(\xi_k)}{\hat{f}_k'(\eta_{k+1})}\right|$.

Proof: Let $g(t) = f(t) - \hat{f}_k(t) - Mw(t)$ where M is chosen so that

(20) $\quad f(t_0) = \hat{f}_k(t_0) + Mw(t_0)$, for some $t_0 \neq \overset{0}{x}_k, \overset{1}{x}_k, \ldots, \overset{n}{x}_k$.

Clearly $g^{(l_j)}(\overset{j}{x}_k) = 0$ for $j = 0,1,\ldots,n$; $l_j = 0,1,\ldots,\gamma_j-1$, since $d(f,\underline{x}_k) = d(\hat{f}_k,\underline{x}_k)$. Also $g(t_0) = 0$, so by the mean value theorem theorem there exists n+1 points $\overset{i}{x}_1 \in co(\overset{i}{x}_k, \overset{i+1}{x}_k)$, $i = 0,1,\ldots,n-1$, and $\overset{n}{x}_1 \in co(\overset{n}{x}_k, t_0)$ such that $g'(\overset{i}{x}_1) = 0$ for $i = 0,1,\ldots,n$. Repeating this process, using points at which g' is zero, we get points where $g^{(2)}$ is zero. If we keep doing this, we eventually see that there exists a $\xi \in co(\overset{0}{x}_k, \overset{1}{x}_k, \ldots, \overset{n}{x}_k, t_0)$ such that $g^{(q)}(\xi) = 0$, where $q = \sum_{j=0}^{n} \gamma_j$. But

$$g^{(q)}(\xi) = f^{(q)}(\xi) - \hat{f}_k^{(q)}(\xi) - Mw^{(q)}(\xi)$$

and $w^{(q)}(\xi) = 1/q!$. Thus $M = \frac{1}{q!}\left(f^{(q)}(\xi) - \hat{f}_k^{(q)}(\xi)\right)$, or from (20)

$$f(t_0) = \hat{f}_k(t_0) + \frac{1}{q!}\left(f^{(q)}(\xi) - \hat{f}_k^{(q)}(\xi)\right) w(t_0).$$

To prove (19), let $t_0 = \alpha(f) = \alpha$, then

(21) $$\hat{f}_k(\alpha) = \frac{-1}{q!}\left(f^{(q)}(\xi_k) - \hat{f}_k^{(q)}(\xi_k)\right) w(\alpha) ,$$

where $\xi_k \in co(\overset{0}{x}_k, \overset{1}{x}_k, \ldots, \overset{n}{x}_k, \alpha)$. Now if $\hat{f}_k$ has a zero α_k,

$$\hat{f}_k(\alpha) = \hat{f}_k'(\eta_{k+1})\ (\alpha-\alpha_k)$$

where $\eta_{k+1} \in co(\alpha,\alpha_k)$. By definition $\hat{\Phi}(d(f,\underline{x}_k)) = x_{k+1} = \alpha_k$, so

$$x_{k+1} - \alpha = -\hat{f}_k(\alpha)/\hat{f}_k'(\eta_{k+1}) \quad .$$

Using (21) we get

$$x_{k+1} - \alpha = \frac{1}{q!}\left[\frac{f^{(q)}(\xi_k) - \hat{f}_k^{(q)}(\xi_k)}{\hat{f}_k'(\eta_{k+1})}\right]\prod_{j=0}^{n}(\alpha - x_k^j)^{\gamma_j}$$

from which (19) follows.

We are now ready to give conditions under which sufficient algorithms are optimal. We shall first consider the case where we are finding roots of functions from R→R and where the data is one-point with memory.

(22) Theorem: Let $\mathcal{D}(\mathcal{F})$ consist of data of the form $d(f,\underline{x}_k) = (x_k,x_{k-1},\ldots,x_{k-n},f(x_k),Df(x_k),\ldots,D^{\gamma_0-1}f(x_k),f(x_{k-1}),\ldots,$ $D^{\gamma_n-1}f(x_{k-n}))$ and let $q = \sum_{j=0}^{n}\gamma_j$. Suppose $f:R\to R$ $|D^q f(x)| \leq M$, and $Df(\alpha(f))$ is invertible for all $f \in \mathcal{F}$ and $x \in T_f$. Let $\hat{\Phi} \in \mathcal{A}(\mathcal{D}(\mathcal{F}))$ be a sufficient algorithm with power $\hat{p}$ defined on the sufficient class $\hat{\mathcal{F}}$ and let Φ be another algorithm in $\mathcal{A}(\mathcal{D}(\mathcal{F}))$ with power p. If there exists a function h_1 [h_2] and a solution sequence $\{x_k\}$ [$\{z_k\}$] on h_1 [h_2] formed by $\hat{\Phi}$ [Φ] such that