SIAM J. SCI. COMPUT.
Vol. 17, No. 5, pp. 1232–1248, September 1996

013

LOCATING AND COMPUTING ALL THE SIMPLE ROOTS AND EXTREMA OF A FUNCTION*

DIMITRIS J. KAVVADIAS†‡ AND MICHAEL N. VRAHATIS†

Abstract. This paper describes and analyzes two algorithms for locating and computing with certainty all the simple roots of a twice continuously differentiable function $f: (a, b) \subset \mathbb{R} \to \mathbb{R}$ and all the extrema of a three times continuously differentiable function in (a, b). The first algorithm locates and computes all the simple roots or all the extrema, while the second one is more efficient in the case where both simple roots and extrema are required.

This paper also gives analytical estimation of the expected complexity of the algorithms based on the distribution of the roots in (a, b). Here only the case of uniform distribution is examined, which is also the approach to be followed when no statistical data are available for the function at hand.

The algorithms have been implemented and tested. Performance information for a well-known Bessel function is reported.

Key words. zeros isolation, Kronecker–Picard theory, topological degree, locating simple roots and extrema, computing simple roots and extrema, combinatorial optimization, zeros identifications, distribution of the roots and extrema, expected algorithms complexity, Bessel functions

AMS subject classifications. 65H05, 68C25

1. Introduction.

It is well known that information concerning all the roots and/or all the extrema of a function

$$f: (a, b) \subset \mathbb{R} \to \mathbb{R} \tag{1.1}$$

is of major importance in many different fields of science and technology.

An immediate method to accomplish this task suggests scanning the interval (a, b) and applying some rootfinding method for each consecutive root. The method that is employed here is heavily based on the knowledge of the total number of roots within (a, b). In order to obtain this information we use results from topological degree theory and especially from the theory of the Kronecker–Picard integral [24, 25]. This theory gives a formula for the computation of the total number of roots of a system of equations within a given region. With this tool in hand one can construct a procedure for the localization and isolation of all the roots by dividing the given region successively and applying the above formula to these subregions until the final domains contain at most one root. To our knowledge, the first method to this end was introduced by Hoenders and Slump [10, 11, 28], who recently reconsidered and applied Kronecker–Picard theory to calculate the total number of simple roots of a system of nonlinear equations as well as to calculate the total number of multiple roots of a single equation of any multiplicity.

Other approaches that have been used successfully to find all solutions of systems of equations as well as the global optimum of a function are based on interval analysis (see, for example, [1, 7, 8, 9, 15, 16, 17]). The corresponding existence tool of these methods is the availability of the range of the function in a given interval, which can be implemented very efficiently using interval arithmetic, though accuracy problems must be resolved. This tool, however, reports with certainty only the negative case, i.e., when no roots are in the interval. The positive case proceeds by subdividing the interval into two halves and employing additional criteria. This case is common when the function has many extrema but few roots in the interval of interest. In addition, this tool provides the existence of a root and not their exact number, which means that more sophisticated decisions (than merely subdividing) cannot be made.

*Received by the editors March 30, 1994; accepted for publication (in revised form) May 15, 1995. This work was partially supported by the EEC ESPRIT Basic Research Action 7141 (ALCOM II).

†Department of Mathematics, University of Patras, GR-261.10 Patras, Greece (vrahatis@math.upatras.gr).

‡Computer Technology Institute, P.O. Box 1122, GR-261.10 Patras, Greece (djk@math.upatras.gr).

In this paper we implement Kronecker–Picard theory and give a method for locating and computing all the simple roots of a twice continuously differentiable function and all the extrema of a three times continuously differentiable function in an interval (a, b). In addition, we give analytical expressions of the total *expected* work needed in achieving the isolation and computation of all the roots of (1.1) under certain assumptions that we present in the sequel. This may prove to be very useful in real life applications since very early estimations of the total running time of the algorithm are available, contrary to other methods which are totally unpredictable. Nevertheless, it is important to notice that our analytical results and techniques also apply to any method that proceeds by repeated subdivisions, giving the expected depth of the subdivision under the same assumptions. Moreover, the main computational burden of our method comes from the need of an integration, a problem that has been studied extensively and for which numerous methods exist.

The rootfinding portion of our method employs a modification of the well-known bisection method. Alternatively, any one of the one-dimensional rootfinding methods (see [23, 20, 21, 26, 2]) can be used. We use the bisection method for several reasons which we explain in §2.2, among which is its known behavior concerning the number of iterations required when we seek the root with a predetermined accuracy.

Here we study the one-dimensional case only. The reason for this restriction is that the corresponding analysis of our method to n dimensions seems to be quite different.

2. Theoretical aspects. Our algorithms are separated in two phases: (1) the phase of the localization and the isolation of all the roots, and (2) the rootfinding phase for the computation of all the roots within a predetermined accuracy.

For the localization phase we use a method of determining the total number of simple roots of a single equation within a given interval. To this end we briefly exploit degree theory for determining the exact number of roots of a single equation by computing the value of topological degree using Kronecker's integral [19, 3, 29, 22, 13] on Picard's extension [24, 25, 10, 28, 11].

For the rootfinding phase we have chosen the bisection method for reasons to be explained later.

2.1. The topological degree for the computation of the total number of roots and extrema. Let us first define the notion of the topological degree. Suppose that the function $F_n = (f_1, \ldots, f_n)\colon \mathcal{D}_n \subset \mathbb{R}^n \to \mathbb{R}^n$ is defined and two times continuously differentiable in a bounded domain $\mathcal{D}_n$ of $\mathbb{R}^n$ with boundary $b(\mathcal{D}_n)$. Suppose further that the roots of the equation $F_n(x) = \Theta_n$ ($\Theta_n = (0, \ldots, 0)$ denotes the origin of $\mathbb{R}^n$) are not located on $b(\mathcal{D}_n)$ and they are simple; i.e., the Jacobian determinant of F_n at these roots is nonzero. Then the *topological degree of F_n at Θ_n relative to $\mathcal{D}_n$* is denoted by $\deg[F_n, \mathcal{D}_n, \Theta_n]$ and can be defined by the following sum:

$$\deg[F_n, \mathcal{D}_n, \Theta_n] = \sum_{x \in F_n^{-1}(\Theta_n)} \operatorname{sgn} J_{F_n}(x), \tag{2.1}$$

where J_{F_n} denotes the determinant of the Jacobian matrix and sgn defines the sign function.

The above definition can be generalized when F_n is only continuous [23]. In this case, Kronecker's theorem [3, 23] states that $F_n(x) = \Theta_n$ has at least one root in $\mathcal{D}_n$ if $\deg[F_n, \mathcal{D}_n, \Theta_n] \neq 0$. Furthermore, if $\mathcal{D}_n = \mathcal{D}_n^1 \cup \mathcal{D}_n^2$ where $\mathcal{D}_n^1$ and $\mathcal{D}_n^2$ have disjoint interiors and $F_n(x) \neq \Theta_n$ for all $x \in b(\mathcal{D}_n^1) \cup b(\mathcal{D}_n^2)$, then the topological degree is additive.

Several methods for the computation of the topological degree have been proposed in the past few years [29, 22, 13, 14, 30, 31, 4]. Also, $\deg[F_n, \mathcal{D}_n, \Theta_n]$ can be represented by the

Kronecker integral, which is closely tied with facts used later and is defined as follows:

$$\deg[F_n, \mathcal{D}_n, \Theta_n] = \frac{\Gamma(n/2)}{2\pi^{n/2}} \int\limits_{b(\mathcal{D}_n)} \int \cdots \int \frac{\sum_{i=1}^{n} A_i dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n}{({f_1}^2 + {f_2}^2 + \ldots + {f_n}^2)^{n/2}}, \tag{2.2}$$

where A_i is defined by the following determinant:

$$A_i = (-1)^{n(i-1)} \begin{vmatrix} F_n & \dfrac{\partial F_n}{\partial x_1} & \cdots & \dfrac{\partial F_n}{\partial x_{i-1}} & \dfrac{\partial F_n}{\partial x_{i+1}} & \cdots & \dfrac{\partial F_n}{\partial x_n} \end{vmatrix}. \tag{2.3}$$

Now, since $\deg[F_n, \mathcal{D}_n, \Theta_n]$ is equal to the number of simple roots of $F_n(x) = \Theta_n$ which give positive Jacobian minus the number of simple roots which give negative Jacobian, then of course the total number $\mathcal{N}^r$ of simple roots of $F_n(x) = \Theta_n$ can be obtained by the value of $\deg[F_n, \mathcal{D}_n, \Theta_n]$ if all these roots give the same Jacobian sign. To this end Picard considered the following extensions of the function F_n and the domain $\mathcal{D}_n$:

$$F_{n+1} = (f_1, \ldots, f_n, f_{n+1}) \colon \mathcal{D}_{n+1} \subset \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}, \tag{2.4}$$

where $f_{n+1} = y\, J_{F_n}$, $\mathbb{R}^{n+1} : x_1, x_2, \ldots, x_n, y$, and $\mathcal{D}_{n+1}$ is the direct product of the domain $\mathcal{D}_n$ with an arbitrary interval of the real y–axis containing the point $y = 0$. Then the roots of the system of equations

$$\begin{aligned} &f_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1, \ldots, n, \\ &y\, J_{F_n}(x_1, x_2, \ldots, x_n) = 0 \end{aligned} \tag{2.5}$$

are the same simple roots of $F_n(x) = \Theta_n$ provided $y = 0$. On the other hand, it is easily seen that the Jacobian of (2.5) is equal to $\left(J_{F_n}(x)\right)^2$, which is always positive. Thus, we conclude that the total number $\mathcal{N}^r$ of roots of $F_n(x) = \Theta_n$ can be given by the following relation:

$$\mathcal{N}^r = \deg[F_{n+1}, \mathcal{D}_{n+1}, \Theta_{n+1}]. \tag{2.6}$$

We consider now the problem of calculating the total number of simple roots of $f(x) = 0$, where $f \colon (a, b) \subset \mathbb{R} \to \mathbb{R}$ is twice continuously differentiable in a predetermined interval (a, b), where a and b are arbitrarily chosen such that $f(a)\, f(b) \neq 0$. According to Picard's extension we define the function $F_2 = (f_1, f_2) \colon \mathcal{P} \subset \mathbb{R}^2 \to \mathbb{R}^2$ and the corresponding system

$$\begin{aligned} f_1(x, y) &= f(x) = 0, \\ f_2(x, y) &= y f'(x) = 0, \end{aligned} \tag{2.7}$$

where the prime denotes differentiation and where $\mathcal{P}$ is an arbitrarily chosen rectangular parallelepiped in the (x, y)–plane given by $a \leq x \leq b$ and $-\gamma \leq y \leq \gamma$ with γ a small positive constant.

Now, since the roots are simple, which means $f'(x) \neq 0$ for $x \in f^{-1}(0)$, it is easily seen that the roots of (2.7) in the above-defined region are the same roots as (1.1). Also, since $J_{F_2} = {f'}^2$, the total number of simple zeros $\mathcal{N}^r$ of the function (1.1) in (a, b) can be given by

$$\mathcal{N}^r = \deg[F_2, \mathcal{P}, \Theta_2]. \tag{2.8}$$

Now, for $n = 2$, by applying (2.1) and using the relations $df_j = \frac{\partial f_j}{\partial x_1} dx_1 + \frac{\partial f_j}{\partial x_2} dx_2$, where $j = 1, 2$, we can easily obtain

$$\mathcal{N}^r = \frac{1}{2\pi} \oint_{b(\mathcal{P})} \frac{f_1 df_2 - f_2 df_1}{{f_1}^2 + {f_2}^2} = \frac{1}{2\pi} \oint_{b(\mathcal{P})} d \arctan\left(\frac{f_2}{f_1}\right). \tag{2.9}$$

Replacing f_1 and f_2 by virtue of (2.7) and performing the integration in (2.9) we finally get

$$\mathcal{N}^r = -\frac{1}{\pi}\left[\gamma\int_a^b \frac{f(x)f''(x) - f'^2(x)}{f^2(x) + \gamma^2 f'^2(x)}\,dx + \arctan\left(\frac{\gamma f'(b)}{f(b)}\right) - \arctan\left(\frac{\gamma f'(a)}{f(a)}\right)\right]. \tag{2.10}$$

It has been shown by Picard [24, 25] that the relation (2.10) is independent of the value of γ.

Of course the total number $\mathcal{N}^e$ of the extrema of $f \in C^3$, i.e., $x \in (a, b)$ such that $f'(x) = 0$, can be obtained by setting in (2.9) $f_1 = f'$ and $f_2 = yf''$.

Remark 2.1. Using (2.10) it is possible that in many applications $\mathcal{N}^r$ can be computed analytically. If not, one can use numerical integration [26].

The Kronecker–Picard integral can also be applied for the determination of the total number of multiple roots [5, 32]. Along these lines, Hoenders and Slump gave in [11] a method for the determination of the total number of multiple roots of a single function. According to their method, if $f\colon (a, b) \subset \mathbb{R} \to \mathbb{R}$ is a k times continuously differentiable function, then the total number of zeros $\mathcal{N}_m^r$ of $f(x) = 0$ with multiplicity m or higher where $m \le k$ can be obtained by the value of the topological degree calculated in a parallelepiped $\mathcal{P}$ by using in (2.7) as f_1 and f_2

$$\begin{aligned} f_1 &= f^2 + \sum_{l=1}^{m}\left(\frac{d^l f}{dx^l}\right)^2, \\ f_2 &= yf_1'. \end{aligned} \tag{2.11}$$

2.2. A modified bisection method. *Notation* 2.1. Throughout this paper the notation $\lceil\cdot\rceil$ refers to the smallest integer not less than the real number quoted; $\ell(I)$ indicates the length of the interval I.

It is well known that a solution of $f(x) = 0$ where the function f is continuous is guaranteed to exist in some interval $[a, b]$ if the following criterion is fulfilled:

$$f(a)\,f(b) \le 0. \tag{2.12}$$

This criterion is known as Bolzano's existence criterion. Instead of Bolzano's criterion one may also use the value of the topological degree of f at origin relative to (a, b), which in this case can be defined as follows:

$$\deg[f, (a, b), 0] = \frac{1}{2}\left(\operatorname{sgn} f(b) - \operatorname{sgn} f(a)\right). \tag{2.13}$$

Now, if $\deg[f, (a, b), 0]$ is not zero, we know with certainty that there is at least one root in (a, b). Note that if $\deg[f, (a, b), 0]$ is not zero, then Bolzano's criterion is fulfilled. The value of $\deg[f, (a, b), 0]$ gives additional information concerning the behavior of the solutions of $f(x) = 0$ in (a, b) relative to the slopes of f [35]. For example, if $\deg[f, (a, b), 0] = 1$, which means that $f(b) > 0$ and $f(a) < 0$, then the number of solutions at points where $f(x)$ has a positive slope exceeds by one the number of solutions at points at which $f(x)$ has a negative slope.

Using the value of the topological degree (or, alternatively, Bolzano's criterion) we are able to calculate a solution of $f(x) = 0$ by bisecting the interval $I_0 = (a, b)$. So we subdivide I_0 into two intervals $(a, c], [c, b)$ where $c = (a + b)/2$ is the midpoint of (a, b) and we keep the subinterval for which the value of the topological degree is not zero relative to itself by checking the information on the boundaries. In this way we always keep at least one solution within a smaller interval. We can continue this procedure in order to approximate a solution until the endpoints of the final subinterval differ from each other by less than a fixed amount. This method is called the *bisection method* and can be generalized to higher dimensions [33, 34].

The main idea in order to locate all the solutions r_j, where $j = 1, \ldots, \mathcal{N}^r$ of $f(x) = 0$ in (a, b), is to subdivide the interval (a, b) and find $\mathcal{N}^r$ subintervals (a_j, b_j) for which the relation (2.12) is fulfilled. Now, for each one subinterval (a_j, b_j) we can apply any method to compute the root which is included in (a_j, b_j). Here we shall use the above-mentioned bisection method which has been modified to the following simplified version described in [33, 34]:

$$x_{i+1} = x_i + \operatorname{sgn} f(x_0) \operatorname{sgn} f(x_i)\, \ell(I_0)/2^{i+1}, \qquad x_0 = a, \qquad i = 0, 1, \ldots . \tag{2.14}$$

The sequence (2.14) converges to a root $r \in (a, b)$ if, for some x_i, $\operatorname{sgn} f(x_0) \operatorname{sgn} f(x_i) = -1$. Also, the number of iterations ν, which are required in obtaining an approximate root r^* such that $|r - r^*| \leq \varepsilon$ for some $\varepsilon \in (0, 1)$, is given by

$$\nu = \lceil \log_2(\ell(I_0)\, \varepsilon^{-1}) \rceil. \tag{2.15}$$

Instead of the iterative formula (2.14) we can also use

$$x_{i+1} = x_i - \operatorname{sgn} f(x_0) \operatorname{sgn} f(x_i)\, \ell(I_0)/2^{i+1}, \qquad x_0 = b, \qquad i = 0, 1, \ldots . \tag{2.16}$$

The bisection method always converges within the given interval (a, b) and it is a global convergence method. Moreover, it has a great advantage since it is optimal; i.e., it possesses asymptotically the best possible rate of convergence [27]. Also, using the relation (2.15) it is easy to have beforehand the number of iterations that are required for the attainment of an approximate root to a predetermined accuracy. Finally, it requires only the algebraic signs of the functions values to be computed, as is evident from (2.14) or (2.16); thus, it can be applied to problems with imprecise function values.

3. The algorithms. The algorithms are a sort of "guided" form of the bisection method. They use the outcome of relation (2.10) in order to isolate the roots or extrema by dividing the initial interval (a, b) into smaller intervals. Algorithm *find_roots*, described below in "pseudo Pascal," first determines the number of roots in the interval by applying (2.10) (step 8) and, if there are more than one, it divides the interval into m_n equal-size subintervals (steps 6 and 7) and proceeds recursively to each of the subintervals. We propose m_n to be equal to the number of roots though there are several issues to be considered here. We discuss this subject later on.

ALGORITHM *find_roots*(a, b, S);
{**comment**: This algorithm locates and computes all the roots or all the extrema of $f(x) = 0$ in (a, b). It exploits (2.10) and (2.14). For (2.10) it requires f, f', f'', and γ while for (2.14) it requires f and ε}

01. procedure roots$(a, b, \mathcal{N}^r)$; {**comment**: adds to set S the $\mathcal{N}^r$ roots of the interval (a, b)}
begin
02. **if** $\mathcal{N}^r = 1$ **then** find the single root r using the bisection (2.14), set $S \longleftarrow S \cup \{r\}$
else
begin
03. $j \longleftarrow 1$; {**comment**: this counts the subintervals $I_j = (a_j, b_j)$}
04. $k \longleftarrow 0$; {**comment**: this counts the computed roots}
05. **while** $k < \mathcal{N}^r$ **do**
begin
06. $a_j \longleftarrow a + (j-1)\frac{b-a}{m_n}$; {**comment**: m_n is the number of subintervals in which we choose to divide (a, b)}
07. $b_j \longleftarrow a + j\frac{b-a}{m_n}$;
08. Find $\mathcal{N}_j^r$, the number of roots in I_j using (2.10);
09. **if** $\mathcal{N}_j^r > 0$ **then** roots$(a_j, b_j, \mathcal{N}_j^r)$;

```
10.                 k ⟵ k + N_j^r;
11.                 j ⟵ j + 1
                  end {while}
              end
          end {roots}
      begin {find_roots}
12.   input a, b; {comment: f(a) f(b) must be nonzero}
13.   S ⟵ ∅; {comment: S is the set of roots in (a, b)}
14.   Find N_0^r, the number of roots in (a, b) using (2.10);
15.   roots(a, b, N_0^r);
16.   output S
      end. {find_roots}
```

Remark 3.1. The case where only the extrema are required can be handled by the same algorithm by replacing the function f by its first derivative f'.

In the case where both roots and extrema are required, we can certainly apply the above method twice for f and f', respectively. But since the extrema lie between consecutive simple roots which are discovered by the first run of *find_roots* (for f), we now choose to divide initially at exactly the points of the roots. We next apply *find_roots* for f' for each subinterval between two consecutive roots. A high-level description of this algorithm follows.

```
ALGORITHM roots_extrema(a, b, S');
{comment: This algorithm locates and computes all the roots and all the extrema of f(x)
in (a, b). It uses (2.10) and (2.14). For (2.10) it requires f, f', f'', f''', and γ while for
(2.14) it requires f, f', and ε}
begin {roots_extrema}
01. Find N^e, the number of extrema in (a, b) using (2.10);
02. Apply algorithm find_roots; Let S = {r_1, ..., r_{N^r}} be its sorted output.
03. Construct the subintervals I_j = (a_j, b_j) = (r_{j-1}, r_j), j = 1, ..., N^r + 1, r_0 = a,
    r_{N^r+1} = b.
04. E ⟵ N^r − 1; {comment: E counts the number of verified extrema}
05. if f'(a) f'(r_1) < 0 then E ⟵ E + 1;
06. if f'(r_{N^r}) f'(b) < 0 then E ⟵ E + 1;
07. if E = N^e then apply bisection (2.14) (using f') in all I_j. Let S' the set of the extrema.
    else
      begin
08.     S' ⟵ ∅;
09.     j ⟵ 1; {comment: this counts the subintervals I_j = (a_j, b_j)}
10.     k ⟵ 0; {comment: this counts the computed extrema}
11.     while k < N^e − E + j − 1 do
            {comment: when k + E − j + 1 = N^e, it is assured that
             only one extremum exists in each remaining interval}
            begin
12.           Apply algorithm find_roots in I_j using f'; Let S'_j be its output;
13.           Set S' ⟵ S' ∪ S'_j; {comment: S'_j will be the set of the extrema in (a_j, b_j)}
14.           k ⟵ k + |S'_j|;
15.           j ⟵ j + 1
            end {while}
16.     Apply bisection (2.14) (using f') in all I_l for l = j + 1, ..., N^r; set S' ⟵ S' ∪ S'_l
      end
17. output S'
end. {roots_extrema}
```

Returning to algorithm *find_roots* and assuming that the number of roots in a subinterval is always available (say it is given by an "oracle"), it is clear that the computational complexity of the algorithm is determined by (1) the total number of calls to the oracle and (2) the iterations required by the $\mathcal{N}^r$ bisection calls which will compute the isolated roots.

4. The expected complexity of the algorithm. In this section we study the expected complexity of algorithm *find_roots*. First, we focus our attention on the number of times the number of roots has to be found in a given interval, i.e., the *average number of oracle calls* as this is the most demanding step of the localization phase. The rootfinding phase is dominated by the *average number of iterations* of the bisection method.

4.1. Preliminaries, definitions, and notations. The study presented here follows certain assumptions:

(i) The size of the problem is the total number of roots of $f(x)$ in (a, b).

(ii) We view each root of $f(x)$ as a random variable having a given distribution in (a, b). All roots are considered as pairwise independent variables having the same distribution. In this paper we assume that the distribution is *uniform*; i.e., it is equally likely for any point in the interval to be a root.

(iii) The study of the expected complexity of the algorithm is *over all possible inputs to the algorithm*, that is, over all possible sets of n points in the interval (a, b) and, consequently, independent of the given function.

Assumption (iii) implies only partial knowledge of the properties of the specific function (the distribution of its roots), which may vary from complete ignorance as to whether the roots lie (our case) to full knowledge of the roots if the properties of the function are known. Hence, the analysis is independent of the specific function which merely serves (through Picard's integral) as an "oracle" that reveals the number of roots in a specific interval. This models real life applications where some physical quantity is of interest whose zeros are statistically independent with known (or suspected) distribution. It is indeed intermediate cases of the form "the roots are expected around the center of the interval" that are really the most interesting since this type of knowledge may "tune up" appropriate algorithms that take into account the additional information and perform better. It is also worthwhile mentioning that the same analysis applies to any method that isolates the roots by repeated subdivisions of the intervals, independently of how the oracle is implemented (a good example is interval analysis techniques for which the main result of §4.2 is a lower bound). In this paper we completely ignore possible properties of $f(x)$ and solve the following combinatorial problem, leaving the more complete study for future research:

Find the expected number of oracle calls that algorithm find_roots *will require in order to isolate all n points which are randomly and independently chosen with uniform distribution.*

In other words, we want to find the expected value $\big($over all possible patterns of appearances of n roots in the interval $I_0 = (a, b)\big)$ of a function H which, given a specific pattern of roots in the interval, returns the number of oracle calls required to isolate each root in a subinterval (a formal definition follows). Since, however, this function is noncontinuous and we want to avoid integrating in subintervals, we adopt a simpler approach *discretizing* the interval $I_0 = (a, b)$ and summing instead of integrating.

We begin by giving a list of symbols and definitions that we shall need later.

Notation 4.1. Let $|S|$ denote the cardinality of the set S. Let $[\cdot]$ denote the integer part of the number quoted. Notice that $[x, y]$ refers to the closed interval with endpoints x and y.

DEFINITION 4.1. *We call resolution δ of the algorithm a small positive real such that if x is a root in (a, b), any point in the interval $\left[x - \frac{\delta}{2}, x + \frac{\delta}{2}\right]$ is considered to be the root x.*

This definition means that as far as the algorithm is concerned, any two roots less that δ apart are considered to be one and, as one, are reported by the oracle in (2.10). We next consider I_0 to be divided into a large number of consecutive subintervals of length δ which

we call *elementary*. Any subdivision of I_0 into small intervals (as stated in steps 6 and 7 of the algorithm, as well as any subdivision thereof) is considered to take place at points that are *integer* multiples of δ. Going one step beyond, we consider (for simplicity reasons) I_0 to be rounded to the smallest integer multiple of δ, say μ, such that $a + \mu\delta \geq b$. Clearly, $\mu = \lceil \frac{b-a}{\delta} \rceil$. Consequently, if m roots are reported by the oracle at step 14 of the algorithm, steps 6 and 7 will divide I_0 with length $\ell(I_0)$ into m intervals I_i, $i = 1, \ldots, m$ such that

$$I_i = \left(a + \delta \left[(i-1)\frac{\ell(I_0)}{m\delta}\right], a + \delta \left[i\frac{\ell(I_0)}{m\delta}\right]\right], \ i = 1, 2, \ldots, m.$$

In the above we adopt the convention that all subintervals be open from the left and closed from the right, thus avoiding a point belonging in more than one subinterval at each level of subdivision. Unifying these conventions, we consider the initial interval I_0 to be $I_0 = (a, a + \mu\delta]$.

The above discretization suggests a more convenient way of representing intervals, namely, as sets of the elementary consecutive subintervals which are included in them. More specifically, consider assigning to each elementary subinterval of I_0 an integer from 1 to μ in increasing order from left to right. This suggests representing I_0 by the set of consecutive integers $T_0 = \{l \in \mathbb{N}: 1 \leq l \leq \mu\}$. The representation of a subinterval I_i is

$$T_i = \left\{l \in T_0: \left[(i-1)\frac{\ell(I_0)}{m\delta}\right] + 1 \leq l \leq \left[i\frac{\ell(I_0)}{m\delta}\right]\right\}.$$

Analogous relations hold for any subinterval of I_0. For our study, T_0 is the sample space with each of its points a candidate of being a root. In the sequel we shall use the term "interval" both for a standard interval I and its corresponding T set when no confusion arises. Similarly, extending Notation 2.1, we denote by $\ell(T)$ the length of the corresponding standard interval I.

In the analysis that follows we shall denote by X a variable representing a set of integers (or, for our purposes, a set of roots). If $X \subset T_l$, X is any set of elementary subintervals, i.e., a set of integers in T_l. In the context described above, the probability of a set X of cardinality k, denoted by Prob$\{X\}$, is the probability of choosing the k elementary subintervals that correspond to the k roots. For example, if the distribution of the roots is uniform, then the probability of a specific X set is $\binom{\mu}{|X|}^{-1}$. Using the above, we are now ready to formally define the function H.

DEFINITION 4.2. *Let S be the set of all subsets of T_0. The function $H: X \in S \to \mathbb{N}$ maps X to the number of oracle calls that the algorithm will do in order to isolate the roots represented by the set X. Similarly, we define the H function relative to an interval T denoted by $H_T(X)$ and which gives the number of oracle calls that the algorithm will do in the specific interval T; that is, in the latter case we only count oracle calls required for roots inside T.*

Remark 4.1. If the interval T_0 is divided in m_n subintervals, T_i, $i = 1, \ldots, m_n$, the number of oracle calls in T_0 is clearly $H_{T_0} = 1 + \sum_{i=1}^{m_n} H_{T_i} - 1 = \sum_{i=1}^{m_n} H_{T_i}$. The "1" comes from the initial oracle call which returns the number of roots in T_0 and the "-1" from the fact that no oracle call is required to obtain the number of roots of the last interval since this can be obtained by subtracting from the total number the sum of the roots in the rest of the $m_n - 1$ subintervals. The same relation holds for the value of the H function of any interval. Note that here we assume that all $m_n - 1$ oracle calls will be required in the while loop of step 5.

4.2. Theoretical results. We now study the expected behavior of our algorithm beginning from the first stage, where the roots are isolated each in an interval by its own. Specifically, we study the *expected number of oracle calls* required for this task as a function of the number

of roots n. Observe that the number of intervals into which we choose to divide can be either constant or a function of n. By n we denote the total number of roots in the given interval (n is the $\mathcal{N}_0^r$ of step 14 of the algorithm *find_roots*.) In any case, deciding the value of m_n is based only on n, and this is the reason why we denoted the number of subdivisions subscribed by n. As stated in Remark 4.1, the approach that follows assumes that all $m_n - 1$ calls are required in an interval which is divided into m_n subintervals. This is equivalent to replacing the while statement of step 5 by the statement **while** $j < m_n$ **do**. This is done for simplicity reasons and it is removed after the next theorem is proved.

THEOREM 4.1. *Suppose that the algorithm find_roots divides any interval T_0 with n roots into m_n subintervals T_i, $i = 1, \ldots, m_n$. Then the expected number of oracle calls required by the algorithm to isolate all n roots is given by the formula*

$$E_H(n) = {m_n}^{1-n} \sum_{k=0}^{n} \binom{n}{k} (m_n - 1)^{n-k} E_H(k) \tag{4.1}$$

or, equivalently,

$$E_H(n) = \sum_{k_1+\cdots+k_{m_n}=n} \frac{n!}{k_1! \cdots k_{m_n}!} {m_n}^{-n} \Big(E_H(k_1) + \cdots + E_H(k_{m_n}) \Big), \tag{4.2}$$

where $E_H(0) = E_H(1) = 1$.

Proof. We shall show first (4.2). Clearly, $E_H(0) = E_H(1) = 1$ since, when no root or a single root is in T_0, then only one oracle call is required. For $n \geq 2$ let X_i, $i = 1, \ldots, m_n$ be the set of roots that lay in interval T_i. Clearly, $|X_1| + \cdots + |X_{m_n}| = n$. The distribution of the corresponding vector of the cardinalities of the X_i's $(|X_1|, \ldots, |X_{m_n}|)$ is polynomial; that is,

$$\text{Prob}\{|X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}\} = \frac{n!}{k_1! \cdots k_{m_n}!} {m_n}^{-n}. \tag{4.3}$$

This is immediate since Prob{some specific root lies in T_i} $= {m_n}^{-1}$ because of the uniform distribution of the roots in (a, b).

Let $\tilde{X} = X_1 \cup X_2 \cup \cdots \cup X_{m_n}$. By Remark 4.1 we have

$$H(\tilde{X}) = H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n}).$$

Taking the expectation [6] of both sides of the above we have

$$\begin{aligned} E\{H(\tilde{X})\} &= E\{H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n})\} \\ &= E\left\{ E\big\{ H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n}) \,\Big|\, |X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n} \big\} \right\}, \end{aligned} \tag{4.4}$$

where the outer expectation is over all sets $\tilde{X} = X_1 \cup \cdots \cup X_{m_n}$ such that $|X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}$ with $k_1 + \cdots + k_{m_n} = n$. Now, the expectation of $H_{T_i}(X_i)$ where X_i runs over all possible choices of sets in T_i with $|X_i| = k_i$ depends only on the cardinality k_i of X_i. We may, therefore, simplify our notation and denote by $E_H(k_i)$ the expectation $E\{H_{T_i}(X_i)\}$. Note that we have also dropped the subscript T_i since the expectation does not depend on the specific interval either, as the distribution is uniform. Using linearity in the inner conditional expectation and the above observation we get

$$E_H(n) = E\left\{ E_H(k_1) + \cdots + E_H(k_{m_n}) \,\Big|\, |X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n} \right\}.$$

Using (4.3) we finally obtain

$$E_H(n) = \sum_{k_1+\cdots+k_{m_n}=n} \frac{n!}{k_1!\cdots k_{m_n}!}\, {m_n}^{-n} \Big(E_H(k_1) + \cdots + E_H(k_{m_n})\Big).$$

In order to show (4.1), observe that the above equation may be written as

$$\begin{aligned} E_H(n) &= \sum_{k=0}^{n} \binom{n}{k} {m_n}^{-k} \sum_{\substack{k_1+\cdots+k_{i-1}\\ +k_{i+1}+\cdots+k_{m_n}=n-k}} \binom{n-k}{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_m} {m_n}^{k-n} E_H(k) \\ &= \sum_{k=0}^{n} {m_n}^{-k} \binom{n}{k} \left(\frac{1}{m_n} + \cdots + \frac{1}{m_n}\right)^{n-k} E_H(k), \end{aligned}$$

where $m_n - 1$ fractions are added. Equation (4.1) now follows easily. □

Remark 4.2. Solving (4.1) with respect to $E_H(n)$ results in the following formula which can be used to obtain numerical values for $E_H(n)$:

$$E_H(n) = \left({m_n}^{n-1} - 1\right)^{-1} \sum_{k=0}^{n-1} \binom{n}{k} (m_n - 1)^{n-k} E_H(k). \tag{4.5}$$

An interesting question is raised as to whether we can remove recursion from (4.1). It turns out that recurrences of the above form are very difficult to handle. We were unable to do so in the general case, that is, when m is a function of n, or even in the case $m = n$, which is particularly interesting. However, when m is constant, we reduced (4.1) to a known recurrence solved in [18] by the use of *binomial transformations.* This is done by finding a recurrence for the sequence $E'_H(k) = E_H(k) - 1,\ k = 0, 1, \ldots$. By employing the techniques described in [18, p. 501] we can show that for $n \geq 2$

$$E'_H(n) = (m-1) \sum_{k=2}^{n} \binom{n}{k} \frac{(-1)^k (k-1) m^{k-1}}{m^{k-1} - 1}. \tag{4.6}$$

Observe that m is referred to without a subscript since the above hold when m is a constant.

We conclude the study of the first phase by examining the complexity of the algorithm *find_roots* as stated originally, that is, when step 5 includes the condition $k < \mathcal{N}^r$. Then we may be able to save several oracle calls if all the roots fall into the first few intervals and, consequently, it is unnecessary to call the oracle for the rest of the intervals. In order to calculate how much we overestimated the expected complexity in Theorem 4.1, observe that in (4.4) the expectation is over every possible $\tilde{X}$ with cardinality n. But if $\tilde{X}$ totally falls in intervals T_1 to T_i, algorithm *find_roots* will not require oracle calls for the last $m_n - i$ intervals. So the expectation was overestimated by $(m_n - i) \times \text{Prob}\{\tilde{X}$ totally falls in intervals T_1 to $T_i\}$. For example, when $\tilde{X}$ is totally included in the first interval, then a term $(\frac{1}{m_n})^n (m_n - 1)$ must be subtracted. Similarly, an additional $(m_n - 2)\big((\frac{2}{m_n})^n - (\frac{1}{m_n})^n\big)$ must be subtracted when $\tilde{X}$ is totally included in the first two intervals. The subtracted $(\frac{1}{m_n})^n$ is the probability of $\tilde{X}$ to fall totally in the first interval and was already counted in the term $(\frac{1}{m_n})^n (m_n - 1)$. Hence, we must decrease (4.1) by

$$\sum_{i=1}^{m_n} \left[\left(\frac{i}{m_n}\right)^n - \left(\frac{i-1}{m_n}\right)^n\right] (m_n - i) = \sum_{i=1}^{m_n - 1} \left(\frac{i}{m_n}\right)^n.$$

We, therefore, have the following theorem.

THEOREM 4.2. *The expected number of oracle calls of algorithm find_roots is given by*

$$E_H(n) = m_n^{1-n} \sum_{k=0}^{n} \binom{n}{k} (m_n - 1)^{n-k} E_H(k) - \sum_{i=1}^{m_n - 1} \left(\frac{i}{m_n} \right)^n . \tag{4.7}$$

Proof. The proof follows immediately from Theorem 4.1 and the above discussion. □

We next proceed to the study of the second phase of the algorithm when all the roots have been isolated, each in its own interval. This second phase consists of a number of bisections that are applied in those intervals. Our task, therefore, requires the definition of a complexity function that will capture the actual work of the second phase, like the H function defined above. A promising approach might be to use the *average length* of the n intervals that are searched by bisection. (Note here that we are talking about the average length of the intervals of a *specific* run of the algorithm and not over every possible run. Should we accept this approach, we must next calculate the *expected average length* of the intervals.) But there is a subtle point here. The actual average case work of the bisection is logarithmically related to the length of the interval in which it is applied (see §2). This means that if we want to estimate the work of the second phase, the average (the arithmetic mean, to be precise) length of the intervals is not the best measure. In fact, it is not difficult to see that the *geometric mean* is a quantity proportional to the total number of iterations required. Seeking for an appropriate complexity measure we give the following definition.

DEFINITION 4.3. *Let $\tilde{X}$ be the set of roots and T any subinterval of T_0. We denote by $SP[T; \tilde{X}]$ the set of subsets of T into which the algorithm will divide T and that include one element of $\tilde{X}$.*

In other words, $\mathrm{SP}[T; \tilde{X}]$ is the set of subintervals of T produced by the algorithm each containing a single root and to which bisection must be applied. Observe now that the total number of iterations IT, involved in phase two, is given by

$$\mathrm{IT} = \sum_{T \in \mathrm{SP}[T_0; \tilde{X}]} \log_2 \left(\ell(T)\, \varepsilon^{-1} \right).$$

The above formula has the disadvantage that it involves the absolute length of the intervals that remains after phase one, which makes it inappropriate for a recursive relation. By a slight modification we get the following relation which, contrary to the above, involves the relative (with respect to the initial interval T_0) lengths of the intervals:

$$\mathrm{IT} = \sum_{T \in \mathrm{SP}[T_0; \tilde{X}]} \log_2 \frac{|T|}{|T_0|} + n \log_2 \left(\ell(T_0)\, \varepsilon^{-1} \right). \tag{4.8}$$

Observe that if we manage to estimate the average value of the sum, then finding the expected number of iterations is a trivial task. Therefore, we focus our attention to this sum which we call the "characteristic complexity function of phase two" and define it as follows.

DEFINITION 4.4. *Let $\tilde{X}$ be a set of roots in a specific instance of the problem. The "characteristic complexity function" of phase two is defined by*

$$B(\tilde{X}) = \sum_{T \in \mathrm{SP}[T_0; \tilde{X}]} \log_2 \frac{|T|}{|T_0|}.$$

Similarly, we define the function B relative to an interval T' and denote it by $B_{T'}$ to be

$$B_{T'}(\tilde{X}) = \sum_{T \in \mathrm{SP}[T'; \tilde{X}]} \log_2 \frac{|T|}{|T'|}.$$

Our next task will be to calculate the expected value of B.

THEOREM 4.3. *Suppose that the algorithm divides an interval T_0 with n roots into m_n subintervals. Then the expected value of the characteristic complexity function is given by the formula*

$$E_B(n) = {m_n}^{1-n} \sum_{k=0}^{n} \binom{n}{k} (m_n - 1)^{n-k} E_B(k) - n \log_2 m_n \tag{4.9}$$

or, equivalently,

$$E_B(n) = \sum_{k_1+\cdots+k_{m_n}=n} \frac{n!}{k_1! \cdots k_{m_n}!} {m_n}^{-n} \left(E_B(k_1) + \cdots + E_B(k_{m_n}) \right) - n \log_2 m_n, \tag{4.10}$$

where $E_B(0) = E_B(1) = 0$.

Proof. First observe that $E_B(0) = 0$ since $\mathrm{SP}[T; \emptyset] = \emptyset$. Also, $E_B(1) = 0$ since when only one root is in an interval, the whole interval must be searched by the bisection. For any $n \geq 2$ let, as in Theorem 4.1, $X_i, i = 1, \ldots, m_n$ be the set of roots in intervals $T_i, i = 1, \ldots, m_n$, respectively. Also, let $\tilde{X}$ be the total set of roots in T_0. Then

$$\begin{aligned} B(\tilde{X}) &= \sum_{T \in \mathrm{SP}[T_0; \tilde{X}]} \log_2 \frac{|T|}{|T_0|} = \sum_{T \in \mathrm{SP}[T_1; X_1]} \log_2 \frac{|T|}{|T_0|} + \cdots + \sum_{T \in \mathrm{SP}[T_{m_n}; X_{m_n}]} \log_2 \frac{|T|}{|T_0|} \\ &= \sum_{T \in \mathrm{SP}[T_1; X_1]} \log_2 \frac{|T|}{|T_1|} - k_1 \log_2 m_n + \cdots + \sum_{T \in \mathrm{SP}[T_{m_n}; X_{m_n}]} \log_2 \frac{|T|}{|T_{m_n}|} - k_{m_n} \log_2 m_n, \end{aligned}$$

where $k_1, \ldots, k_{m_n}$ are the numbers of roots in intervals $T_1, \ldots, T_{m_n}$, respectively. Consequently, under the above assumptions

$$B(\tilde{X}) = B_{T_1}(X_1) + B_{T_2}(X_2) + \cdots + B_{T_{m_n}}(X_{m_n}) - n \log_2 m_n. \tag{4.11}$$

The key observation in (4.11) is, once again, that the expectations of the B functions are independent of the intervals in which they are applied and depend only on the cardinalities of the corresponding X set. Moreover, the subtracted term $n \log_2 m_n$ is independent of the X sets. Hence, by taking the expectations of both sides of (4.11) we may apply the same line of thought as in Theorem 4.1 and obtain (4.10). Proving (4.9) is then completely analogous to proving relation (4.1). □

Remark 4.3. Solving (4.9) with respect to $E_B(n)$ we get the following formula which may be used to obtain numerical values for $E_B(n)$:

$$E_B(n) = \frac{1}{{m_n}^{n-1} - 1} \sum_{k=0}^{n-1} \binom{n}{k} (m_n - 1)^{n-k} E_B(k) - n {m_n}^{n-1} \log_2 m_n. \tag{4.12}$$

COROLLARY 4.4. *The expected number of iterations required by the algorithm in order to compute all n roots is given by the formula*

$$\mathrm{IT} = E_B(n) + n \log_2 \left(\ell(T_0)\, \varepsilon^{-1} \right). \tag{4.13}$$

Proof. The proof follows from (4.8) and Theorem 4.3. $E_B(n)$ must be computed from (4.12). □

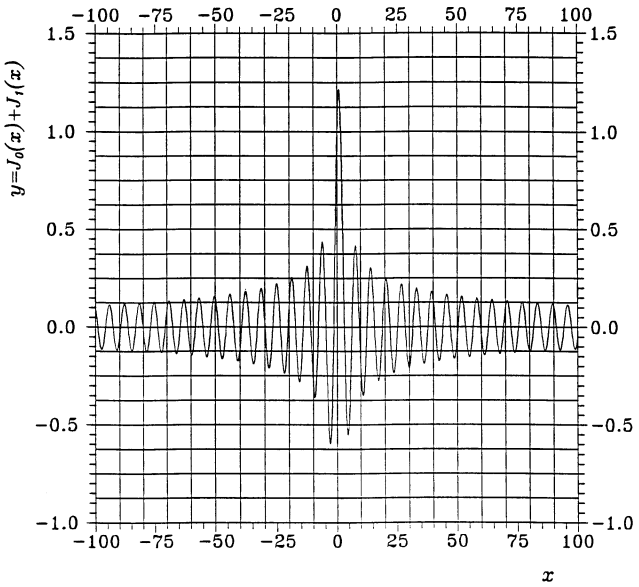

FIG. 1. *The function* (5.1) *for* $c = 0$, $a = -100$, *and* $b = 100$.

5. An example and numerical applications. We illustrate our method and analysis on a parametric problem based on a well-known function also studied in [10]:

$$f(x) = J_0(x) + J_1(x) + c, \tag{5.1}$$

where J_0 and J_1 indicate the zero-order and first-order Bessel functions [26], respectively, and c a constant, where

$$J_n(x) = \sum_{k=0}^{\infty} (-1)^k \frac{(x/2)^{2k+n}}{k!(n+k)!}, \quad n = 0, 1, \ldots . \tag{5.2}$$

Function (5.1) was selected as this and related functions attract the attention of several researchers. Moreover, it possesses some nice properties which make it especially suitable for our purposes. It has a large number of roots which means that arbitrarily large rootfinding problems may be generated, restricted only by the boundaries of the interval. Also, the roots of the function are within a small neighborhood almost equally spaced, which, in turn, results in a performance of the algorithm close to that predicted by the analytical estimations. Finally, as it is obvious from Figure 1, by varying the constant c we can, in effect, raise or lower the function relative to the x–axis and, consequently, "move" the roots toward one end of the interval and study the results in the performance of the algorithm. We applied the algorithm *find_roots* for various intervals in order to compute the corresponding roots. The behavior of the algorithm for the extrema is analogous.

Using the relations

$$\begin{aligned} J_0'(x) &= -J_1(x), \\ J_1'(x) &= J_0(x) - \frac{1}{x} J_1(x), \end{aligned} \tag{5.3}$$

we are able to find the derivatives used in (2.10) as functions of $J_0(x)$ and $J_1(x)$ which, in turn, may be computed using, for example, the routines of [26].

TABLE 1
Computer runs and behavior of the algorithm find_roots for various instances of the problem.

Function	a	b	$\mathcal{N}^r$	OC	IT	EOC	EIT
$J_0(x)+J_1(x)$	-1000	1000	636	636	26712	1027.9	25438.4
$J_0(x)+J_1(x)$	-100	100	63	63	2646	101.1	2521.5
$J_0(x)+J_1(x)$	0	100	31	31	1302	49.3	1241.9
$J_0(x)+J_1(x)-0.125$	-100	100	50	67	2194	80.0	2018.0
$J_0(x)+J_1(x)-0.125$	0	100	25	32	1205	39.6	1009.4
$J_0(x)+J_1(x)-0.125$	0	200	25	40	1611	39.6	1034.4
$J_0(x)+J_1(x)-0.125$	0	300	25	43	1760	39.6	1049.1
$J_0(x)+J_1(x)-0.15$	-100	100	34	50	1522	54.2	1392.4
$J_0(x)+J_1(x)-0.15$	0	100	17	25	933	26.7	696.2
$J_0(x)+J_1(x)-0.15$	0	200	17	29	1202	26.7	713.2
$J_0(x)+J_1(x)-0.15$	0	300	17	31	1302	26.7	723.1

Using bisection for this kind of problem has the advantage that only signs need to be computed, which can be achieved by considering relatively few terms of (5.2).

The results for locating and computing all the zeros are summarized in Table 1. The columns labeled a and b indicate the endpoints of the searched interval; the columns labeled $\mathcal{N}^r$, OC, and IT show the number of computed roots within (a, b), the number of oracle calls, and the number of iterations required by the algorithm *find_roots*, respectively. The two final columns EOC and EIT indicate the values of (4.7) and (4.13) for the same parameters.

As it is evident from Table 1, the predictions from (4.7) and (4.13) are closer to the actual performance in the cases where the pattern of the roots has a certain degree of "asymmetry" that captures the possible variations from uniformity.

6. Discussion and open problems. Let us now focus our attention on the results of §4 and discuss the expected behavior of the algorithm as described by (4.7) and (4.12). The first equation describes the behavior of phase one of the algorithm as a function of n, the number of roots. In Figure 2 we plot (4.7) for various values of m_n, the number of subintervals in which we divide. It is clear that the expected number of calls decreases when we decrease the number of subintervals with a minimum reached for $m_n = 2$. An interesting choice seems to be $m_n = n$ since the curve for this case remains close to the curve of $m_n = 2$.

The reverse holds for the number of iterations of phase two which is plotted in Figure 3. There we plot (4.12) since this is actually the expected number of iterations "saved" by this method (this is represented by the negative sign of the E_B values) compared with n bisections on the interval T_0. When the actual number of iterations is desired, then the term $n \log_2 \left(\ell(T_0)\, \varepsilon^{-1}\right)$ must be added. It is clear, therefore, that what we save must be compared with this last term. Figure 3 shows that the saved iterations grow roughly like $-10n$ (for $m_n = 2$). The added term depends on the quantity $\ell(T_0)\, \varepsilon^{-1}$. Assuming a value of 10^{12} reasonable for this term, i.e., 12 significant digits (this corresponds to 12 decimal digits for the normalized case $\ell(T_0) = 1$), we see that the added term grows like $40n$. Hence, we save about 25% of the iterations. Less demanding accuracy, for example, for $\ell(T_0)\, \varepsilon^{-1} = 10^6$, results in a savings of about 50%. (At this point one may wonder whether the negative part of (4.13) exceeds, in absolute value, the positive part for suitable $\ell(T_0)$ and ε. This is not the case since the analysis above presumes that each subinterval is greater than ε, the desired accuracy. Hence, (4.13) holds for sufficiently small ε.)

It is clear, therefore, that the total complexity of the algorithm is a combination of the complexities of phases one and two. A central issue is the choice of m_n: phase one requires small m_n, if possible two, but phase two requires large m_n. A reasonable choice (which is also the one proposed in the algorithms) seems to be $m_n = n$.

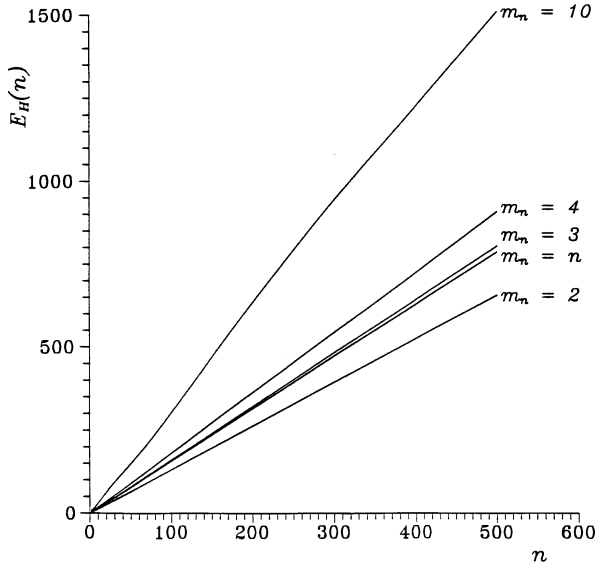

FIG. 2. *Expected number of oracle calls by virtue of* (4.7) *for various values of* m_n.

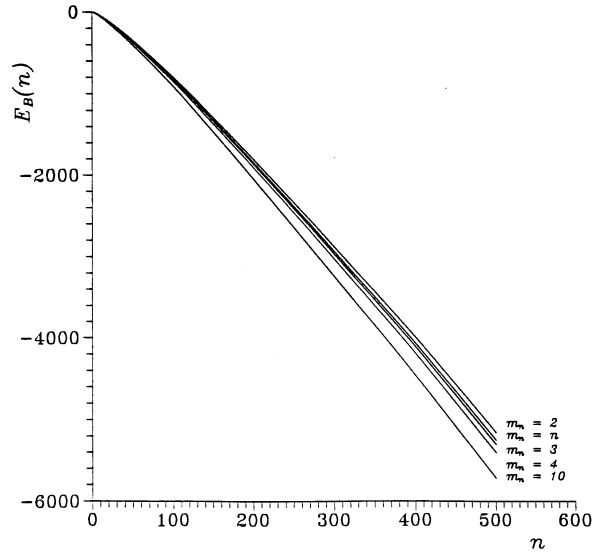

FIG. 3. *Expected number of iterations saved by virtue of* (4.12) *for various values of* m_n.

Every "real life" estimation, however, heavily depends on the way the "oracle" is implemented. The central part of (2.10) is clearly the integral. If the integral can be implemented in a nontime-consuming way (for example, when the analytical form can be found or an efficient numerical method can be applied), then phase one will be less demanding and we must concentrate on phase two by increasing the m_n value. If, however, this is not the case, then probably small values of m_n would be preferred. In any case, the integral must be computed with an error of, at most, 0.5 since it is known that (2.10) returns an integer. This suggests that relatively few function evaluations will be necessary. And, most importantly, the implementation of the oracles for the subintervals can take into account the function evaluations

that have already been computed for the oracle of the divided interval. This last observation suggests that numerical values should be stored and reused for the oracles of higher levels of subdivision. Such an approach could dramatically reduce the total number of function evaluations for the oracles implementation. The full exploitation of this idea and the trade-off between phases one and two will be presented in a future publication.

Several other open problems remain to be examined in different directions. A first direction is to apply the method for more general distributions of the roots. This will, first of all, broaden the class of problems for which the analytical estimations apply. Next, one may apply the same techniques and derive complexity bounds for other methods on the same problem along the lines of Theorems 4.1 and 4.3. But we also feel that there are several things to be examined here in the algorithm itself. For example, if we expect the roots to be concentrated around the middle of the interval, then clearly more refined subdivisions must take place there, while close to the boundaries the subdivision can be sparser. That is, we believe that the algorithm itself can be guided if the distribution of the roots is known, in order to improve its performance. Results on this direction will be reported in [12].

Furthermore, preliminary investigations suggest that the algorithms can be generalized to higher dimensions to locate and compute with certainty all the simple roots of equations $F(x) = (0, 0, \ldots, 0)$ where $F = (f_1, f_2, \ldots, f_n)\colon (a, b)^n \subset \mathbb{R}^n \to \mathbb{R}^n$, as well as all the extrema of functions $f\colon (a, b)^n \subset \mathbb{R}^n \to \mathbb{R}$.

Finally, we would like to point out the possibilities of efficient parallelization of the algorithm, thus exploiting the tremendous capabilities of modern parallel computers. This opens a totally new subject where all the discussed problems should be reexamined in the light of massive parallelism.

Acknowledgments. The authors wish to thank Stavros Kourouklis for useful discussions that led to an alternative proof of Theorem 4.1 and Dimitris Sotiropoulos for programming assistance.

REFERENCES

[1] G. ALEFELD AND J. HERZBERGER, *Introduction to Interval Computations*, Academic Press, New York, 1983.
[2] G. ALEFELD, F. A. POTRA, AND YIXUN SHI, *On enclosing simple roots of nonlinear equations*, Math. Comp., 61 (1993), pp. 733–744.
[3] P. ALEXANDROFF AND H. HOPF, *Topologie*, Springer-Verlag, Berlin, 1935; reprinted Chelsea, New York, 1965.
[4] T. BOULT AND K. SIKORSKI, *An optimal complexity algorithm for computing the topological degree in two dimensions*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 686–698.
[5] A. DAVIDOGLOU, *Sur le nombre des racines communes à plusieurs équations*, Compt. Rend. hebd., 133 (1901), pp. 784–786 and pp. 860–863.
[6] W. FELLER, *An Introduction to Probability Theory and Its Applications*, Vol. I, John Wiley, New York, 1968.
[7] E. R. HANSEN, *A globally convergent interval method for computing and bounding real roots*, BIT, 18 (1987), pp. 415–424.
[8] ———, *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.
[9] E. R. HANSEN AND G. W. WALSTER, *Nonlinear equations and optimization*, Comput. Math. Appl., 25 (1993), pp. 125–145.
[10] B. J. HOENDERS AND C. H. SLUMP, *On the calculation of the exact number of zeros of a set of equations*, Computing, 30 (1983), pp. 137–147.
[11] ———, *On the determination of the number and multiplicity of zeros of a function*, Computing, 47 (1992), pp. 323–336.
[12] D. J. KAVVADIAS AND M. N. VRAHATIS, *Locating and computing all roots of functions with arbitrarily distributed roots*, manuscript.
[13] R. B. KEARFOTT, *Computing the Degree of Maps and a Generalized Method of Bisection*, Ph.D. thesis, Department of Mathematics, University of Utah, Salt Lake City, UT, 1977.
[14] ———, *An efficient degree–computation method for a generalized method of bisection*, Numer. Math., 32 (1979), pp. 109–127.
[15] ———, *Some tests of generalized bisection*, ACM Trans. Math. Software, 13 (1987), pp. 197–220.

[16] R. B. KEARFOTT AND M. NOVOA III, *INTBIS: A portable interval Newton/bisection package*, ACM Trans. Math. Software, 16 (1990), pp. 152–157.
[17] R. B. KEARFOTT, *Interval arithmetic techniques in the computational solution of nonlinear systems of equations: Introduction, examples, and comparisons*, Lectures in Appl. Math., 26 (1990), pp. 337–357.
[18] D. E. KNUTH, *The Art of Computer Programming, Vol*. 3: *Sorting and Searching*, Addison–Wesley, Reading, MA, 1973.
[19] L. KRONECKER, *Werke*, Vol. 1, Teubner, Leipzig, 1895.
[20] D. LE, *An efficient derivative–free method for solving nonlinear equations*, ACM Trans. Math. Software, 11 (1985), pp. 250–262.
[21] ———, *Three new rapidly convergent algorithms for finding a zero of a function*, SIAM J. Sci. Statist. Comput., 16 (1985), pp. 193–208.
[22] T. O'NEIL AND J. THOMAS, *The calculation of the topological degree by quadrature*, SIAM J. Numer. Anal., 12 (1975), pp. 673–680.
[23] J. M. ORTEGA AND W. C. RHEINBOLT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
[24] E. PICARD, *Sur le nombre des racines communes à plusieurs équations simultanées*, J. Math. Pures Appl. (4^e série), 8 (1892), pp. 5–24.
[25] ———, *Traité d'analyse*, 3rd ed., Gauthier–Villars, Paris, 1922.
[26] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND W. T. VETTERLING, *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1992.
[27] K. SIKORSKI, *Bisection is optimal*, Numer. Math., 40 (1982), pp. 111–117.
[28] C. H. SLUMP AND B. J. HOENDERS, *The determination of the location of the global maximum of a function in the presence of several local extrema*, IEEE Trans. Inform. Theory, 31 (1985), pp. 490–497.
[29] F. STENGER, *Computing the topological degree of a mapping in* $\mathbb{R}^n$, Numer. Math., 25 (1975), pp. 23–38.
[30] M. STYNES, *An Algorithm for the Numerical Calculation of the Degree of a Mapping*, Ph.D. thesis, Department of Mathematics, Oregon State University, Corvallis, OR, 1977.
[31] ———, *An algorithm for numerical calculation of topological degree*, Appl. Anal., 9 (1979), pp. 63–77.
[32] G. TZITZÉICA, *Sur le nombre des racines communes à plusieurs équations*, Compt. Rend. hebd., 133 (1901), pp. 918–920.
[33] M. N. VRAHATIS, *Solving systems of nonlinear equations using the nonzero value of the topological degree*, ACM Trans. Math. Software, 14 (1988), pp. 312–329.
[34] ———, *CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations*, ACM Trans. Math. Software, 14 (1988), pp. 330–336.
[35] ———, *A short proof and a generalization of Miranda's existence theorem*, Proc. Amer. Math. Soc., 107 (1989), pp. 701–703.