SIAM J. Sci. Comput. © 1999 Society for Industrial and Applied Mathematics
Vol. 21, No. 3, pp. 954–969

LOCATING AND COMPUTING ARBITRARILY DISTRIBUTED ZEROS[*]

D. J. KAVVADIAS[†], F. S. MAKRI[†], AND M. N. VRAHATIS[†]

Abstract. The problem of locating and computing with certainty all the simple roots of a twice continuously differentiable function $f\colon [a,b] \subset \mathbb{R} \to \mathbb{R}$ is studied when some additional information on the distribution of the roots in the interval is available. The framework is the one proposed by [*SIAM J. Sci. Comput.*, 17 (1996), pp. 1232–1248], where only the uniform case was examined. This paper settles some of the problems posed there and generalizes some of its results by considering an arbitrary distribution of the roots in $[a,b]$. The theoretical results are accompanied by simulations in a number of problems of various size.

Key words. zeros isolation, Kronecker–Picard theory, topological degree, locating simple roots, computing simple roots, zeros identifications, bisection method, distribution of the roots, expected complexity of algorithms

AMS subject classifications. 65H05, 65Y20

PII. S1064827598333806

1. Introduction.

Many problems in different areas of science, such as mechanics, physical sciences, statistics, operations research, etc., are reduced to the problem of finding all the roots or the extrema of a function in a given interval. The importance of the problem has attracted the attention of many research efforts and as a result many different approaches to the problem exist. We briefly mention here the deflation techniques used for the calculation of further solutions [3] and more recently interval analysis based methods (see, e.g., [6, 7, 14, 15, 16, 17]) and the method described in [10].

Evaluating the performance of an algorithm usually involves experimental observations in known problem instances drawn from the literature and/or randomly constructed ones. Few analytical estimations of widely accepted performance measures exist and these mostly concern the worst-case behavior of an algorithm [2, 20]. The worst-case behavior, however, can sometimes be very conservative for real-life applications, while the expected behavior seems to be a more natural and informative measure. Contrary to the apparent importance of the expected behavior of an algorithm, however, even fewer analytical results exist [4]. One reason for this might be the difficulty inherent in such a study and the lack of a suitable framework.

In [10] a framework for the study of the expected complexity of the problem of finding with certainty all the simple roots of a function was presented and some results were shown for the case when the roots are uniformly distributed in the interval. While this models the situation where no information of the distribution of the roots is available, it is clearly a severe restriction of the general problem, as sometimes additional information about concentrations of roots is available. Such information can in turn be modeled as a mathematical probability distribution and thus the clustering behavior can be taken into account.

[*]Received by the editors September 24, 1998; accepted for publication (in revised form) January 12, 1999; published electronically November 23, 1999. This work was partially supported by the Greek General Secretariat of Research and Technology (ΠΕΝΕΔ program 850).

http://www.siam.org/journals/sisc/21-3/33380.html

[†]Department of Mathematics, University of Patras Artificial Intelligence Research Center, University of Patras, GR–261.10 Patras, Greece (djk@math.upatras.gr, makri@math.upatras.gr, vrahatis@math.upatras.gr).

In this paper we study a generalization of an algorithm proposed in [10] that takes into account the probability distribution of the roots. This algorithm is a good representative of a more general class of algorithms which proceed by subdividing the interval into smaller intervals until each root is isolated in a subinterval. Classical methods (e.g., bisection or Newton) are then applied to actually identify the root. All different competing methods draw some kind of information from an "oracle" on the existence or not of roots in the interval. Depending on the oracle this information may vary from complete knowledge of the exact number of roots in the interval to a mere indication of the existence or not of root(s).

The former method was used, for example, in [10], while the latter is applied in interval analysis based methods. As might be expected, more informative oracles are, in general, also more computationally expensive. A central issue that arises is if this additional information can be exploited to speedup the overall performance of the method. We mention here that the results and the techniques developed in this paper also have an impact on the above-mentioned class of methods.

In [10] the expected number of oracle calls required by the algorithm in order to isolate n roots was given under the assumption that the roots are uniformly distributed in the given interval. Here we abandon the uniformity assumption and show that the expected number of oracle calls of the modified algorithm is given by the same formula (4.1) of [10] for which we study its growth rate and show that it is $O(n \log n)$. In addition, for the case of arbitrary distribution, we give an upper bound of the expected complexity of the second phase.

Theoretical results are accompanied by numerical ones from which very useful pieces of information are drawn and new problems are posed for further research.

In section 2 some basic theory of the algorithm described in section 3 is given. In section 4 we give the theoretical study of the behavior of the algorithm. The last section includes some simulations and a discussion on the results and poses some open problems.

2. Preliminaries.

A simple oracle on the existence of a solution of $f(x) = 0$ in some interval (a, b), where the function f is continuous in $[a, b]$, is the following criterion:

$$f(a)\,f(b) < 0 \quad \text{or} \quad \operatorname{sgn} f(a) \operatorname{sgn} f(b) = -1, \tag{2.1}$$

where sgn is the well-known three valued sign function. This criterion is known as Bolzano's existence criterion. Note that this oracle may introduce a one-sided error, i.e., a positive response means that at least one root exists but a negative response may correspond to the existence of an even number of simple roots.

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:

$$\deg[f, (a, b), 0] = \frac{1}{2}\Big(\operatorname{sgn} f(b) - \operatorname{sgn} f(a)\Big). \tag{2.2}$$

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 [24]. 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.

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, e.g., [1, 5, 6, 8, 12, 13, 14, 15, 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 is error-free when it responds “no,” while a “yes” answer may be misleading. The positive case proceeds by subdividing the interval into two halfs and employing additional criteria.

A more complicated but completely informative oracle which gives the exact number of roots $\mathcal{N}^r$ is based on topological degree theory using Kronecker’s integral on a Picard’s extension [9, 18]. For the computation of the topological degree, see, e.g., [2, 11, 21, 22]. This oracle was used in [10] as part of the first phase of an algorithm for the isolation of all the simple roots of a function $f(x)$ in an interval (a, b) and returns the number of roots using the formula

$$\mathcal{N}^r = -\frac{\gamma}{\pi}\int_a^b \frac{f(x)f''(x) - f'^2(x)}{f^2(x) + \gamma^2 f'^2(x)}\,dx + \frac{1}{\pi}\arctan\left(\frac{\gamma\,[f(a)f'(b) - f(b)f'(a)]}{f(a)f(b) + \gamma^2 f'(a)f'(b)}\right), \tag{2.3}$$

where γ is a small positive constant. Note that $\mathcal{N}^r$ was shown to be independent of the value of γ [18].

The algorithm of [10] uses, in a second phase, bisection in order to compute the roots. Specifically, it uses the following simplified version described in [23]:

$$x_{i+1} = x_i + c\,\mathrm{sgn} f(x_i)/2^{i+1}, \qquad c = \mathrm{sgn} f(a)\,(b-a), \qquad i = 0, 1, \ldots . \tag{2.4}$$

The sequence (2.4) converges to a root $r \in (a, b)$ if for some x_i, $\mathrm{sgn} f(x_0)\,\mathrm{sgn} f(x_i) = -1$, for $i = 1, 2, \ldots$. Also, the number of iterations ν, which are required in obtaining an approximate root r^* such that $|r - r^*| \le \varepsilon$ for some $\varepsilon \in (0, 1)$, is given by

$$\nu = \lceil \log(b-a)\,\varepsilon^{-1} \rceil, \tag{2.5}$$

where the logarithm in the above relation and also in the rest of the paper is taken with base two. Instead of the iterative formula (2.4) we can also use the following one:

$$x_{i+1} = x_i - c\,\mathrm{sgn} f(x_i)/2^{i+1}, \qquad c = \mathrm{sgn} f(b)\,(b-a), \qquad i = 0, 1, \ldots . \tag{2.6}$$

The reason for choosing the bisection method is that it 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 [20]. Also, using the relation (2.5) 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 function values to be computed, as is evident from (2.4) or (2.6); thus it can be applied to problems with imprecise function values. As a consequence for problems where the function value follows as a result of an infinite series (e.g., Bessel or Airy functions) it can be shown [25, 26] that the sign stabilizes after a relatively small number of terms of the series and the calculations can be sped up considerably.

3. The algorithm.

Here we present a generalization of Algorithm *find_roots* of [10] as a *model* for studying the effects of different subdivisions of the interval. To allow complete decision abilities we have employed the Kronecker–Picard integral described in the previous section as our oracle. As in [10] the algorithm decides the number of subdivisions after it has discovered the number of roots in the interval. Its generality comes from the fact that its decisions can be quite arbitrary: the algorithm may choose to take full advantage of the (known) number of roots and divide, for example, into a number of intervals that equal the number of roots or it may ignore most of the information and divide into a fixed number of subintervals, for example, two. The only restriction is that the decision must always be the same for a specific number of roots n. This is shown in the algorithm by denoting the number of subdivisions by m_n. The value of m_n is fixed once n is given but it can be quite arbitrary.

The main difference from [10] comes from the way the interval is subdivided at each step into smaller subintervals. In *find_roots* the subdivision was done into subintervals of equal length. In the algorithm that follows (which we call *generalized_find_roots*) the subdivision creates subintervals $I_j, j = 1, 2, \ldots, m_n$, of equal probability; that is, $\int_{I_j} \varphi(x)dx = m_n^{-1}$, where $\varphi(x)$ is the probability density function of the roots. Note that subdividing into subintervals of equal length under uniform distribution is, in effect, subdivision into subintervals of equal probability and thus the current approach generalizes [10].

Algorithm *generalized_find_roots*$(a, b, \mathcal{S})$.

{**comment**: This algorithm locates and computes all the roots of $f(x) = 0$ in (a, b). It exploits (2.3) and (2.4). For (2.3) it requires f, f', f'', γ, and φ while for (2.4) it requires f and ε. The roots are stored in set $\mathcal{S}$.}

```
01. procedure roots($a, b, \mathcal{N}^r$);
      {comment: adds to set $\mathcal{S}$ 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 (2.4), set $\mathcal{S} \longleftarrow \mathcal{S} \cup \{r\}$
          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.               $a_j \longleftarrow a + \sum_{i=1}^{j-1} \ell(I_i)$;
                  {comment: $I_j$ is the $j$th interval (from the left end) for
                   which $\int_{I_j} \varphi(x)dx = m_n^{-1}$; $\ell(I_i)$ is the length of interval $I_i$;
                   $m_n$ is the number of subintervals in which we choose to
                   divide $(a, b)$, see above}
07.               $b_j \longleftarrow a_j + \ell(I_j)$;
08.               Find $\mathcal{N}^{r,j}$, the number of roots in $I_j$ using (2.3);
09.               if $\mathcal{N}^{r,j} > 0$ then roots($a_j, b_j, \mathcal{N}^{r,j}$);
10.               $k \longleftarrow k + \mathcal{N}^{r,j}$;
11.               $j \longleftarrow j + 1$
                end {while}
            end
        end {roots}
    begin {generalized_find_roots}
12.   input $a, b$;  {comment: $f(a) \cdot f(b)$ must be nonzero}
13.   $\mathcal{S} \longleftarrow \emptyset$;  {comment: $\mathcal{S}$ is the set of roots in $(a, b)$}
```

```
14.    Find $\mathcal{N}^{r,0}$, the number of roots in $(a,b)$ using (2.3);
15.    *roots*$(a,b,\mathcal{N}^{r,0})$;
16.    output $\mathcal{S}$
  **end.** {*generalized_find_roots*}
```

In the above description of the algorithm, the number of roots is revealed by the use of (2.3). However, as we have already mentioned in the introduction, we shall henceforth view this step as a "black box" (an oracle), the actual implementation of which is omitted. Thus the computational complexity of the Algorithm *generalized_find_roots* is determined by (i) the total number of calls to the oracle and (ii) the iterations required by the $\mathcal{N}^r$ bisection calls which will compute the isolated roots.

4. A study on the behavior of the algorithm.

In this section we study the expected complexity of Algorithm *generalized_find_roots*. First, we focus our attention on the number of times the algorithm calls the oracle, as this is the most demanding step of the localization phase. We give a recursive formula for the expected number of oracle calls under a given distribution, which turns out to be the same as the expected number of oracle calls of Algorithm *find_roots* of [10]. We then bound the growth rate of this formula and show that it grows asymptotically like $O(n \log n)$. We end our study with a bound on the expected number of iterations under an arbitrary distribution.

4.1. Definitions and notations.

We next describe the framework on which our analysis is based. The basic assumptions and definitions come from [10].

We consider the number of roots n to be given and we view the n roots as independent identically distributed random variables that assume values in the interval (a,b). The formula for the expected number of oracle calls will, in general, depend on n which represents the size of the problem and on the given distribution φ.

A basic assumption which reflects the limited accuracy of any real computer model is that we consider our interval to be composed of a large number of small consecutive subintervals of length δ which we call *elementary*. Any elementary interval may contain only one root. Notice that this also guarantees that the algorithm always terminates after a finite number of subdivisions.

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 (where μ is the number of elementary subintervals of I_0) in increasing order from left to right. This suggests representing I_0 by the set of consecutive integers $T_0 = \{l \in \mathbb{N} \colon 1 \leq l \leq \mu\}$. Similarly, any subinterval I_i of I_0 is represented as a set of consecutive integers, the smallest one being the integer assigned to the elementary subinterval of its left end and the largest the integer assigned to its right end. In the rest of the paper we shall use the term "interval" both for a standard interval I and the corresponding T set when no confusion arises.

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 $X \subset T_l$, where T_l is a set of integers representing interval I_l, X is any set of elementary subintervals, i.e., a set of integers in T_l.

The next definition formally introduces the main object of our study.

DEFINITION 4.1. *Let P be the set of all subsets of T_0. The function $H \colon X \in P \to \mathbb{N}$ 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 function H* 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 count only oracle calls required for roots inside T.*

4.2. Theoretical results. In this section we prove the theoretical results of the paper. Our first theorem gives the expected number of oracle calls in terms of a recursive formula. The analysis is based on a simplifying assumption, namely, that all $m_n - 1$ oracle 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**. A way to remove this assumption which also applies here is described in [10].

THEOREM 4.1. *The expected number of oracle calls $\mathrm{E}_H^{\varphi}(n)$ required by Algorithm generalized_find_roots to isolate all n roots in an interval is independent of the distribution φ of the roots. That is, by using the expected number of oracle calls for the uniform case (Theorem* 4.1 *of* [10]) *we have*

$$\mathrm{E}_H^{\varphi}(n) = {m_n}^{1-n} \sum_{k=0}^{n} \binom{n}{k} (m_n - 1)^{n-k} \mathrm{E}_H^{\varphi}(k), \tag{4.1}$$

where $\mathrm{E}_H^{\varphi}(0) = \mathrm{E}_H^{\varphi}(1) = 1$ and m_n is the number of subintervals into which the algorithm divides an interval with n roots.

Proof. Let us denote the m_n subintervals into which the interval T_0 is divided by $T_i, i = 1, \dots, m_n$. We shall show the independence from the distribution φ by induction on the number of roots n. That is, we shall show that $\mathrm{E}_H^{\varphi}(n) = \mathrm{E}_H(n)$ for every n, where by $\mathrm{E}_H(n)$ we denote the expectation of the oracle calls when the distribution is uniform.

Clearly $\mathrm{E}_H^{\varphi}(0) = \mathrm{E}_H^{\varphi}(1) = 1$, since when no root or a single root is in T_0 then only one oracle call is required, independently of the distribution of roots.

For the induction hypothesis, assume that $\mathrm{E}_H^{\varphi}(j) = \mathrm{E}_H(j)$ for all $j \le n - 1$. For $n \ge 2$, let $X_i, i = 1, \dots, m_n$ 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, $(|X_1|, \dots, |X_{m_n}|)$ is polynomial, that is,

$$\mathrm{Prob}\{|X_1| = k_1, \dots, |X_{m_n}| = k_{m_n}\} = \frac{n!}{k_1! \cdots k_{m_n}!} {m_n}^{-n}. \tag{4.2}$$

This is immediate since the algorithm subdivides the interval T_0 into subintervals of equal probability and therefore Prob{some specific root lies in T_i} $= \int_{T_i} \varphi(x)dx = {m_n}^{-1}, i = 1, \dots, m_n$.

Let $\tilde{X} = X_1 \cup X_2 \cup \cdots \cup X_{m_n}$. Note that the total number of oracle calls in an interval equals the sum of oracle calls in the subintervals into which it is divided. This follows from the fact that one oracle call is required for the interval, while no oracle call is required for the last of its subintervals, as the number of roots there can be determined by subtracting from the total number of roots the number of roots in the other subintervals. Hence

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

Taking the expectation of both sides of the above when the distribution is φ and when it is uniform we must show

$$\begin{aligned} &\mathrm{E}^{\varphi}\{H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n})\} \\ &\qquad = \mathrm{E}\{H_{T_1}(X_1) + H_{T_2}(X_2) + \cdots + H_{T_{m_n}}(X_{m_n})\} \end{aligned}$$

or, equivalently

$$\mathrm{E}^{\varphi}\left\{\mathrm{E}^{\varphi}\big\{H_{T_1}(X_1)+H_{T_2}(X_2)+\cdots+H_{T_{m_n}}(X_{m_n})\,\Big|\,|X_1|=k_1,\ldots,|X_{m_n}|=k_{m_n}\big\}\right\}$$
$$=\mathrm{E}\left\{\mathrm{E}\big\{H_{T_1}(X_1)+H_{T_2}(X_2)+\cdots+H_{T_{m_n}}(X_{m_n})\,\Big|\,|X_1|=k_1,\ldots,|X_{m_n}|=k_{m_n}\big\}\right\},$$

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 $k_1+\cdots+k_{m_n}=n$. (Notice that for simplicity, we use the same notation for the T_i's and the X_i's in the left-hand and right-hand sides.) Every term in the sum of the expectation corresponds to a solution of the equation $k_1+\cdots+k_{m_n}=n$. By the induction hypothesis in every solution into which all k_i's are less or equal to $n-1$, the expected number of oracle calls is independent of the distribution and hence depends only on the cardinality of roots in a specific interval. Thus these terms in the left-hand side are equal to the corresponding terms in the right-hand side. What we are left with is a number of solutions of equation $k_1+\cdots+k_{m_n}=n$ (m_n to be specific) in which a single k_i is equal to n and all others are zero. The probability of each of these cases is m_n^{-n}. Hence the expectation of the left-hand side has the term $m_n^{-n}[\mathrm{E}^{\varphi}_{H_{T_1}}(n)+\cdots+\mathrm{E}^{\varphi}_{H_{T_{m_n}}}(n)]$ and the expectation of the right-hand side has the term $m_n^{-n}[\mathrm{E}_{H_{T_1}}(n)+\cdots+\mathrm{E}_{H_{T_{m_n}}}(n)]$. These terms correspond to the events where all n roots lie in a single subinterval. In such a case the algorithm will once again subdivide the interval into m_n subsubintervals. We may therefore substitute each $\mathrm{E}^{\varphi}_{H_{T_i}}(n)$ of the left-hand side (resp., $\mathrm{E}_{H_{T_i}}(n)$ of the right-hand side) following the same procedure as above and end up with m_n^2 terms, call them $\mathrm{E}^{\varphi}_{H_{T_{i_1 i_2}}}(n)$, $1\le i_1,i_2\le m_n$ (resp., $\mathrm{E}_{H_{T_{i_1 i_2}}}(n)$), where $\mathrm{E}^{\varphi}_{H_{T_{i_1 i_2}}}(n)$ denotes the expected number of oracle calls in the i_2-subsubinterval of the i_1-subinterval in the case where all n roots happen to fall into this subsubinterval. The probability of these events is m_n^{-2n}. Continuing in the same manner after k levels of subdivision, the corresponding terms are $m_n^{-kn}\sum_{1\le i_1,\ldots,i_k\le m_n}\mathrm{E}^{\varphi}_{H_{T_{i_1\cdots i_k}}}(n)$ (resp., $m_n^{-kn}\sum_{1\le i_1,\ldots,i_k\le m_n}\mathrm{E}_{H_{T_{i_1\cdots i_k}}}(n)$). This process continues until there is no subinterval containing all n roots, in which case we invoke the induction hypothesis and we are done. Notice that this will necessarily happen because for some k large enough, the corresponding subintervals will become smaller than the minimum required to contain all n roots. (Recall that we have assumed that each elementary subinterval δ may contain only one root and therefore no interval smaller than $n\delta$ may contain all n roots.) Therefore $\mathrm{E}^{\varphi}_H(n)=\mathrm{E}_H(n)$ for all n. Now using Theorem 4.1 of [10] we get relation (4.1). ◻

The independence of the expected number of oracle calls from the distribution suggests a more careful study of relation (4.1). We therefore study the growth rate of $E_H(n)$ with n under the assumption that m_n is constant or, more generally, when m_n is allowed to vary with n but always remains less than a constant value $m_{\max}$.

THEOREM 4.2. *The expected value of oracle calls required by Algorithm generalized_find_roots in order to isolate n roots ($n\ge 2$) is $O(n\log n)$ when m_n is bounded from above by a constant value $m_{\max}$.*

Proof. We show that $\mathrm{E}_H(n)\le c\,n\log(n+1)+1$ for some positive constant c that depends on $m_{\max}$. Notice that we retain the notation m_n, since m_n is allowed to depend on n for the theorem to hold, as long as it remains less than or equal to a maximum constant value. The proof is by induction on the number of roots n. For

$n = 0$ and $n = 1$ the bound holds and thus also the induction basis. Also for $n = 2$ some algebraic manipulation yields the value $m_2 + 1$ and thus the theorem also holds, provided that $c \geq \frac{m_2}{2\log 3}$.

Assume that $\mathrm{E}_H(k) \leq c\,k\log(k+1) + 1$ for all $k = 3, \ldots, n-1$. We shall show that it holds for $k = n$.

By Theorem 4.1 we have

$$\begin{aligned}\mathrm{E}_H(n) &= m_n^{1-n}\sum_{k=0}^{n}\binom{n}{k}(m_n-1)^{n-k}\,\mathrm{E}_H(k)\\ &= \frac{m_n^n}{m_n^{n-1}-1}\sum_{k=0}^{n-1}\binom{n}{k}\frac{1}{m_n^k}\left(1-\frac{1}{m_n}\right)^{n-k}\ \mathrm{E}_H(k)\\ &\leq \frac{m_n^n}{m_n^{n-1}-1}\sum_{k=0}^{n-1}p_k(c\,k\log(k+1)+1),\end{aligned}$$

where $p_k = \binom{n}{k}\frac{1}{m_n^k}\left(1-\frac{1}{m_n}\right)^{n-k}$. Thus,

$$\begin{aligned}\mathrm{E}_H(n) &\leq c\,\frac{m_n^n}{m_n^{n-1}-1}\log\prod_{k=0}^{n-1}(k+1)^{kp_k} + \frac{m_n^n}{m_n^{n-1}-1}\sum_{k=0}^{n-1}p_k\\ &\leq c\,\frac{m_n^n}{m_n^{n-1}-1}\log\left(\frac{\sum_{k=0}^{n-1}(k+1)kp_k}{\sum_{k=0}^{n-1}kp_k}\right)^{\sum_{k=0}^{n-1}kp_k} + \frac{m_n^n}{m_n^{n-1}-1}\left(1-\frac{1}{m_n^n}\right)\end{aligned}$$

by using the inequality of geometric and arithmetic means and the fact that $\sum_{k=0}^{n}p_k = 1$. Therefore we have

$$\begin{aligned}\mathrm{E}_H(n) &\leq c\,\frac{m_n^n}{m_n^{n-1}-1}\log\left(\frac{\sum_{k=0}^{n-1}k^2p_k}{\sum_{k=0}^{n-1}kp_k}+1\right)^{\sum_{k=0}^{n-1}kp_k} + \frac{m_n^n-1}{m_n^{n-1}-1}\\ &\leq c\,n\log\left(\frac{\frac{n}{m_n}\left(1-\frac{1}{m_n}\right)+\frac{n^2}{m_n^2}-\frac{n^2}{m_n^n}}{\frac{n\left(m_n^{n-1}-1\right)}{m_n^n}}+1\right)+m_n+1,\end{aligned}$$

where we have used that $\sum_{k=0}^{n}kp_k = \frac{n}{m_n}$ and also that $\sum_{k=0}^{n}k^2p_k = \frac{n}{m_n}\left(1-\frac{1}{m_n}\right)+\frac{n^2}{m_n^2}$. Notice now that the quantity under the logarithm is less than or equal to $\frac{n}{m_n}+2$. We have thus arrived at

$$\mathrm{E}_H(n) \leq c\,n\log\left(\frac{n}{m_n}+2\right)+m_n+1.$$

The proof now rests on showing that $c\,n\log(\frac{n}{m_n}+2)+m_n+1 \leq c\,n\log(n+1)+1$ or, equivalently, that

$$c\,n\log\frac{n+2m_n}{m_n(n+1)}+m_n \leq 0.$$

To see this, first observe that the quantity under the logarithm assumes its maximum value for $m_n = 2$ and $n = 3$ (the case for $n = 2$ has been examined), namely, 7/8.

Since the logarithm is negative for this value, the left-hand side is maximum for the smallest value of n, that is, 3. Hence, for $c \geq -m_{\max}/[3\log(7/8)] \geq -m_n/[3\log(7/8)]$, the inequality holds. Thus the theorem is proved. $\square$

In the above proof we have used a loose lower bound for the value of c. More precise calculations may reduce this bound but this is insignificant since the dominating term is the $n \log n$. In any case, smaller values of m_n result in smaller lower bounds for c.

The following definition is useful for the analysis of the bisection phase of the algorithm.

DEFINITION 4.2. *Let $\tilde{X}$ be the set of roots and T any subinterval of T_0. We denote by* $\mathrm{SP}[T;\tilde{X}]$ *the set of subintervals of T produced by steps* 6 *and* 7 *of Algorithm generalized_find_roots that include only one element of $\tilde{X}$.*

Using the above definition, the total number of iterations IT involved in the bisection phase is given by

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

The above formula involves $\ell(T)$ for $T \in \mathrm{SP}[T_0;\tilde{X}]$ and ε, i.e., the absolute lengths of the remaining intervals and the predefined accuracy of the algorithm. 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:

$$\mathrm{IT} = \sum_{T \in \mathrm{SP}[T_0;\tilde{X}]} \log\frac{|T|}{|T_0|} + n\log\left(\ell(T_0)\,\varepsilon^{-1}\right). \tag{4.3}$$

The second term depends on the absolute length of the original interval T_0 and the accuracy ε and is independent of the specific pattern of roots $\tilde{X}$. Therefore the expected value of the iterations IT equals this quantity plus the expected value of the sum. This sum was called the "characteristic complexity function of phase two" $B(\tilde{X})$ in [10] and, in essence, represents how many fewer iterations the algorithm will do with input $\tilde{X}$ compared to the number of iterations of n bisection runs in the original interval T_0. Notice that this sum is a negative number. We shall also need the "characteristic complexity function" relative to an interval T' which is given by

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

THEOREM 4.3. *Let the interval T_0 contain n roots which are distributed according to a given distribution φ and let m_n be the number of subintervals that Algorithm generalized_find_roots divides an interval with n roots. Then the expected value of the characteristic complexity function of phase two, $\mathrm{E}_B^{\varphi}(n)$, of Algorithm generalized_find_roots, is less than or equal to the expected value of the characteristic complexity function when the distribution of the roots is uniform. That is,*

$$\mathrm{E}_B^{\varphi}(n) \leq \mathrm{E}_B(n),$$

where

$$\mathrm{E}_B(n) = \sum_{k_1+\cdots+k_{m_n}=n} \frac{n!}{k_1!\cdots k_{m_n}!}\, {m_n}^{-n}\left(\mathrm{E}_B(k_1)+\cdots+\mathrm{E}_B(k_{m_n})\right) - n\log m_n, \tag{4.4}$$

with $\mathrm{E}_B(0) = \mathrm{E}_B(1) = 0$.

Proof. Notice that by Theorem 4.3 of [10] the right-hand side of (4.4) is the expected value of the characteristic complexity function, $\mathrm{E}_B(n)$ of Algorithm *generalized_find_roots*, when the distribution of the roots is uniform, since in this case the Algorithm *generalized_find_roots* behaves as Algorithm *find_roots* of [10] by always dividing in subintervals of equal length.

In order to prove the theorem, we use induction on the number of roots n.

For the basis of the induction we observe that for any distribution φ, $\mathrm{E}_B^{\varphi}(0) = 0$ since $\mathrm{SP}[T;\emptyset] = \emptyset$. Also $\mathrm{E}_B^{\varphi}(1) = 0$ since when only one root is in an interval, the whole interval must be searched by the bisection.

For the induction step, assume that $\mathrm{E}_B^{\varphi}(\nu) \le \mathrm{E}_B(\nu)$ holds for all $\nu \le n-1$.

For any $n \ge 2$, as in Theorem 4.1 let $X_i, i = 1, \dots, m_n$, be the set of roots in intervals $T_i, i = 1, \dots, m_n$, respectively. Also let $\tilde{X}$ be the total set of roots in T_0. Then

$$B(\tilde{X}) = \sum_{T\in \mathrm{SP}[T_0;\tilde{X}]} \log\frac{|T|}{|T_0|} = \sum_{T\in \mathrm{SP}[T_1;X_1]} \log\frac{|T|}{|T_0|} + \cdots + \sum_{T\in \mathrm{SP}[T_{m_n};X_{m_n}]} \log\frac{|T|}{|T_0|}.$$

For notational simplicity, we denote by λ_i the ratio $\frac{|T_i|}{|T_0|}$. Thus, we get

$$\begin{aligned} B(\tilde{X}) &= \sum_{T\in \mathrm{SP}[T_1;X_1]} \log\frac{|T|\lambda_1}{|T_1|} + \cdots + \sum_{T\in \mathrm{SP}[T_{m_n};X_{m_n}]} \log\frac{|T|\lambda_{m_n}}{|T_{m_n}|} \\ &= \sum_{T\in \mathrm{SP}[T_1;X_1]} \log\frac{|T|}{|T_1|} + \cdots + \sum_{T\in \mathrm{SP}[T_{m_n};X_{m_n}]} \log\frac{|T|}{|T_{m_n}|} \\ &\quad + k_1 \log\lambda_1 + \cdots + k_{m_n}\log\lambda_{m_n}, \end{aligned}$$

where $k_1, \dots, k_{m_n}$ are the numbers of roots in intervals $T_1, \dots, T_{m_n}$, respectively.

Consequently, under the above assumptions

$$B(\tilde{X}) = B_{T_1}(X_1) + B_{T_2}(X_2) + \cdots + B_{T_{m_n}}(X_{m_n}) + Y, \tag{4.5}$$

where

$$Y = k_1 \log\lambda_1 + \cdots + k_{m_n}\log\lambda_{m_n}. \tag{4.6}$$

We first show that for the expectation of Y, $\mathrm{E}(Y)$ the following inequality holds:

$$\mathrm{E}(Y) \le -n\log m_n. \tag{4.7}$$

To this end we have

$$\begin{aligned} \mathrm{E}(Y) &= \sum_{k_1+\cdots+k_{m_n}=n} (k_1\log\lambda_1 + \cdots + k_{m_n}\log\lambda_{m_n}) \binom{n}{k_1,\dots,k_{m_n}} \frac{1}{{m_n}^n} \\ &= \sum_{i=1}^{m_n} \log\lambda_i \sum_{k_1+\cdots+k_{m_n}=n} k_i \frac{n!}{k_1!\cdots k_{m_n}!}\,\frac{1}{{m_n}^n} \\ &= \frac{n}{m_n}\sum_{i=1}^{m_n}\log\lambda_i \sum_{\substack{k_1+\cdots+(k_i-1)+\\ k_{i+1}+\cdots+k_{m_n}=n-1}} \frac{(n-1)!}{k_1!\cdots(k_i-1)!\cdots k_{m_n}!}\,\frac{1}{{m_n}^{n-1}} \\ &= n\log\left(\lambda_1\cdots\lambda_{m_n}\right)^{1/m_n}. \end{aligned}$$

Now, since the geometric mean is always less than or equal to the arithmetic mean, we get

$$\mathrm{E}(Y) \le n \log \frac{\lambda_1 + \cdots + \lambda_{m_n}}{m_n} = n \log \frac{1}{m_n}, \tag{4.8}$$

where the equality holds for $\lambda_1 = \cdots = \lambda_{m_n}$.

The proof now rests on showing that

$$\mathrm{E}^{\varphi}\{B_{T_1}(X_1) + \cdots + B_{T_{m_n}}(X_{m_n})\} \le \mathrm{E}\{B_{T_1}(X_1) + \cdots + B_{T_{m_n}}(X_{m_n})\} \tag{4.9}$$

or, equivalently,

$$\mathrm{E}^{\varphi}\left\{\mathrm{E}^{\varphi}(B(X_1)) + \cdots + \mathrm{E}^{\varphi}(B(X_{m_n})) \,\Big|\, |X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}\right\}$$
$$\le \mathrm{E}\left\{\mathrm{E}(B(X_1)) + \cdots + \mathrm{E}(B(X_{m_n})) \,\Big|\, |X_1| = k_1, \ldots, |X_{m_n}| = k_{m_n}\right\},$$

where E^{φ} is the expectation under φ distribution and 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 $k_1 + \cdots + k_{m_n} = n$. As in Theorem 4.1 we have used the same notation for the T_i's and X_i's of the left-hand and right-hand sides.

For the solutions of $k_1 + \cdots + k_{m_n} = n$ such that all k_is are less than or equal to $n-1$, the terms $\mathrm{E}^{\varphi}\{B(X_i) \,|\, |X_i| = k_i\} = \mathrm{E}_B(k_i)$ are less than or equal to the corresponding terms of the right-hand side by the induction hypothesis. The remaining m_n solutions correspond to the cases where all n roots lie in only one of the m_n subintervals. Each one of these events has probability m_n^{-n} to occur. These m_n solutions contribute the term $m_n^{-n}[\mathrm{E}^{\varphi}_{B_{T_1}}(n) + \cdots + \mathrm{E}^{\varphi}_{B_{T_{m_n}}}(n)]$ to the left-hand side. This is similar for the right-hand side.

In these cases the algorithm will again subdivide each T_i to m_n new subintervals. Repeating the same procedure as before, we substitute each $\mathrm{E}^{\varphi}_{B_{T_i}}(n)$ by a sum of terms, some of which are less than or equal to the corresponding terms of the right-hand side by the induction hypothesis and by relation (4.7) while the remaining are $m_n^{-2n}\sum_{1 \le i_1, i_2 \le m_n} \mathrm{E}^{\varphi}_{B_{T_{i_1 i_2}}}(n)$, where $T_{i_1 i_2}$ denotes the i_2 subsubinterval of the i_1 subinterval. Continuing in the same manner after k substitutions we are left with the sum $m_n^{-kn}\sum_{1 \le i_1, \ldots, i_k \le m_n} \mathrm{E}^{\varphi}_{B_{T_{i_1 \cdots i_k}}}(n)$ in the left-hand side and the term $m_n^{-kn} m_n^k \mathrm{E}_B(n)$ in the right-hand side. For these last terms the inequality also holds by the arguments used in the end of the proof of Theorem 4.1. Thus, the theorem is proved. $\square$

5. Simulations, discussion, and further research. In order to understand the results of the previous sections a number of computer simulations are presented in Tables 1–3. The simulations were conducted for a number of problem sizes n and different choices of the number m in which we subdivided an interval. Notice that for simplicity we have chosen to subdivide an interval into a constant number of subintervals, i.e., m_n was independent from n and this is why in the tables m appears without the subscript n. The only exception to this was the value $\mathcal{N}^{r,j}$ for m_n, i.e., the subdivision of any interval I_j is done into a number of subintervals equal to the number of the roots that lie in I_j. After m and n were set, a problem instance was constructed. That is, n points were randomly chosen in the interval $(0,1)$ with the distribution of interest. For this purpose we have employed the random number generator of [19]. Our experiments involved only the uniform and Gaussian

Table 1

Performance of Algorithm generalized_find_roots for various problem sizes with uniform distribution.

		Theory		Simulation			
n	m_n	EOC	EB	EOC	EB	VOC	VB
10	$\mathcal{N}^{r,j}$	16.6	−47.5	16.6	−47.5	3.0	6.6
	2	14.4	−45.8	14.4	−45.8	2.9	6.6
	3	17.8	−48.8	17.6	−48.7	3.4	6.7
	10	32.6	−56.6	32.1	−56.8	7.8	7.7
	50	75.1	−66.0	74.7	−66.0	28.4	9.2
20	$\mathcal{N}^{r,j}$	33.6	−116.0	33.6	−116.0	4.3	9.3
	2	28.8	−112.4	28.8	−112.4	4.1	9.3
	3	35.6	−118.2	35.3	−118.2	4.8	9.5
	10	69.7	−136.8	68.7	−136.9	10.8	10.0
	50	154.9	−149.7	154.1	−149.8	47.2	15.6
50	$\mathcal{N}^{r,j}$	84.8	−357.3	84.8	−357.4	6.8	14.7
	2	72.1	−348.1	72.2	−348.1	6.5	14.7
	3	89.0	−362.7	88.1	−362.7	7.6	15.1
	10	164.1	−404.7	162.1	−404.7	17.3	17.3
	50	534.7	−465.1	531.7	−465.2	72.3	21.3
100	$\mathcal{N}^{r,j}$	170.0	−815.5	170.0	−815.5	9.6	20.9
	2	144.3	−796.9	144.2	−796.9	9.2	20.9
	3	178.1	−826.2	176.3	−826.2	10.7	21.4
	10	329.7	−908.7	324.6	−908.6	24.8	24.5
	50	1219.4	−1074.8	1214.3	−1074.6	96.3	22.6
500	$\mathcal{N}^{r,j}$	852.2	−5242.0	851.9	−5241.6	21.3	46.4
	2	721.4	−5148.5	721.2	−5148.2	20.4	46.2
	3	890.2	−5294.6	881.4	−5294.5	23.7	47.7
	10	1642.7	−5714.7	1622.2	−5714.5	54.9	54.5
	50	3852.5	−6165.9	3837.0	−6165.2	206.2	66.2

distribution with different values of mean μ and standard deviation σ. (Notice that the specific values of σ of the Gaussian distribution were such that the "tails" of the distribution outside $(0, 1)$ were negligible.) These n points represented the roots of a hypothetical function. Then Algorithm *generalized_find_roots* was applied with the constructed instance as an input while the oracle for an arbitrary interval was a simple counting of the points that lay in the given interval. For each pair of n and m, 1000 instances were constructed and the number of oracle calls and the value of the complexity function B were counted for each of the 1000 runs. Recall that B is the sum of the logarithms of the absolute lengths of the intervals into which the roots were isolated. But since the original interval was taken to be of length one, B is actually how many fewer iterations were required compared to n bisection runs in the same interval. Then the expected number of oracle calls (EOC) and the expected value of B (EB) were calculated and exhibited in the tables as the average of these 1000 values.

In Table 1 the performance of Algorithm *generalized_find_roots* for uniformly distributed roots is summarized. Notice that this case is exactly the case of Algorithm *find_roots* of [10] since when the roots are uniformly distributed *generalized_find_roots* reduces to *find_roots*. For verification, Table 1 also includes the EOC and EB as calculated by (4.1) and the right-hand side of relation (4.4). Slight differences between the experimental and theoretically expected are due to the limited number of experiments and unavoidable imperfections of the random number generator. Further information on the behavior of the algorithm is reported in the last two columns of Table 1, in which the variance of the oracle calls (VOC) and of B (VB) are included.

Table 2 summarizes a number of experiments of Algorithm *generalized_find_roots* to Gaussian distributed roots with various choices of mean and standard deviation.

TABLE 2
Algorithm generalized_find_roots applied to Gaussian distributed roots.

	(μ, σ)	$\left(\frac{1}{2}, \frac{1}{8}\right)$		$\left(\frac{2}{5}, \frac{1}{10}\right)$		$\left(\frac{1}{5}, \frac{1}{20}\right)$		$\left(\frac{2}{25}, \frac{1}{50}\right)$		$\left(\frac{1}{25}, \frac{1}{100}\right)$	
n	m_n	EOC	EB	EOC	EB	EOC	EB	EOC	EB	EOC	EB
10	$\mathcal{N}^{r,j}$	16.6	−56.0	16.6	−58.8	16.6	−67.9	16.6	−80.3	16.6	−89.7
	2	14.5	−53.9	14.5	−56.6	14.5	−65.4	14.5	−77.5	14.5	−86.7
	3	17.6	−56.9	17.6	−59.7	17.6	−68.7	17.6	−81.0	17.6	−90.4
	10	31.9	−65.4	31.9	−68.3	31.9	−77.6	31.9	−90.2	31.9	−99.7
	50	74.5	−75.2	74.5	−78.2	74.5	−87.9	74.5	−100.9	74.5	−110.7
20	$\mathcal{N}^{r,j}$	33.5	−133.4	33.5	−139.4	33.5	−158.4	33.5	−183.9	33.5	−203.3
	2	28.7	−129.3	28.7	−135.1	28.7	−153.8	28.7	−179.1	28.7	−198.2
	3	35.0	−135.3	35.0	−141.2	35.0	−160.2	35.0	−185.6	35.0	−205.0
	10	68.0	−154.3	68.0	−160.4	68.0	−179.7	68.0	−205.4	68.0	−225.0
	50	151.1	−167.2	151.0	−173.3	151.0	−192.9	151.0	−218.9	151.0	−238.6
50	$\mathcal{N}^{r,j}$	84.2	−402.8	84.2	−418.3	84.2	−467.3	84.2	−532.6	84.2	−582.0
	2	72.0	−393.6	71.9	−408.9	71.9	−457.5	71.9	−522.4	71.9	−571.6
	3	87.8	−408.3	87.8	−423.8	87.8	−472.7	87.8	−537.8	87.8	−587.1
	10	161.6	−450.7	161.7	−466.4	161.7	−515.7	161.7	−581.2	161.7	−630.8
	50	528.5	−510.4	528.1	−526.0	528.1	−575.3	528.1	−640.8	528.1	−690.4
100	$\mathcal{N}^{r,j}$	169.7	−909.1	169.6	−940.6	169.6	−1039.5	169.6	−1170.8	169.6	−1270.2
	2	144.0	−890.2	143.9	−921.3	143.9	−1020.0	143.9	−1150.9	143.9	−1250.2
	3	175.7	−918.9	175.7	−950.4	175.7	−1049.2	175.7	−1180.4	175.7	−1279.7
	10	323.3	−1002.1	323.1	−1033.6	323.1	−1132.8	323.1	−1264.3	323.1	−1363.8
	50	1216.3	−1168.8	1215.6	−1200.6	1215.6	−1300.0	1215.6	−1431.7	1215.6	−1531.4
500	$\mathcal{N}^{r,j}$	853.3	−5721.2	853.1	−5880.9	853.1	−6379.8	853.1	−7039.8	853.1	−7539.2
	2	721.2	−5625.4	721.6	−5786.3	721.6	−6284.8	721.6	−6944.5	721.6	−7443.7
	3	881.9	−5772.3	881.4	−5931.7	881.4	−6430.4	881.4	−7090.4	881.4	−7589.7
	10	1624.7	−6194.6	1624.7	−6355.7	1624.7	−6855.0	1624.7	−7515.3	1624.7	−8014.9
	50	3841.8	−6643.9	3841.5	−6804.8	3841.5	−7304.4	3841.5	−7965.1	3841.5	−8464.9

TABLE 3
Algorithm find_roots applied to Gaussian distributed roots.

	(μ, σ)	$\left(\frac{1}{2}, \frac{1}{8}\right)$		$\left(\frac{2}{5}, \frac{1}{10}\right)$		$\left(\frac{1}{5}, \frac{1}{20}\right)$		$\left(\frac{2}{25}, \frac{1}{50}\right)$		$\left(\frac{1}{25}, \frac{1}{100}\right)$	
n	m_n	EOC	EB	EOC	EB	EOC	EB	EOC	EB	EOC	EB
10	$\mathcal{N}^{r,j}$	18.7	−57.1	18.4	−60.2	18.3	−70.4	20.5	−82.7	19.4	−93.4
	2	16.2	−55.6	16.0	−58.0	17.0	−68.0	18.2	−81.6	19.2	−91.6
	3	18.5	−57.6	20.0	−61.4	19.7	−71.5	20.7	−84.2	22.6	−94.9
	10	36.1	−67.5	35.7	−70.6	34.6	−80.0	37.4	−92.9	36.7	−103.8
	50	90.6	−75.6	93.1	−79.2	109.9	−92.8	115.0	−108.5	98.6	−116.1
20	$\mathcal{N}^{r,j}$	36.9	−135.1	35.9	−141.3	34.9	−161.2	35.4	−187.3	40.0	−207.4
	2	30.3	−131.4	31.1	−137.6	32.1	−157.6	32.8	−184.2	33.8	−204.2
	3	36.6	−136.7	37.7	−143.7	37.1	−163.4	38.4	−189.3	40.1	−210.2
	10	67.3	−154.2	66.9	−160.4	67.1	−180.2	71.0	−208.0	67.9	−226.8
	50	208.9	−177.3	219.2	−186.3	232.0	−211.6	190.0	−232.0	159.1	−224.4
50	$\mathcal{N}^{r,j}$	92.8	−405.6	89.8	−421.2	85.7	−470.8	85.8	−537.2	88.0	−587.5
	2	73.4	−395.6	74.8	−412.2	75.8	−462.2	75.9	−527.5	76.9	−577.5
	3	90.4	−410.7	90.4	−426.7	90.2	−476.1	92.1	−542.4	92.9	−592.9
	10	167.8	−453.7	169.5	−470.8	168.8	−520.2	166.3	−584.6	170.5	−636.9
	50	575.8	−530.1	559.9	−545.3	467.8	−580.7	403.7	−628.2	447.2	−679.6
100	$\mathcal{N}^{r,j}$	187.5	−912.1	180.6	−943.1	171.1	−1042.4	169.6	−1174.8	171.3	−1275.4
	2	145.8	−892.3	146.6	−924.7	147.6	−1024.7	148.1	−1155.9	149.1	−1255.9
	3	178.3	−921.8	178.4	−953.9	178.6	−1053.3	180.6	−1185.9	180.6	−1286.1
	10	339.1	−1012.0	336.9	−1043.2	328.4	−1137.0	338.7	−1274.9	337.9	−1375.4
	50	997.2	−1141.7	934.9	−1162.3	812.1	−1231.1	904.9	−1365.1	1087.1	−1498.7
500	$\mathcal{N}^{r,j}$	963.9	−5724.1	992.8	−5879.4	882.6	−6375.9	845.6	−7038.0	846.3	−7539.3
	2	722.4	−5625.5	723.1	−5786.4	724.1	−6286.4	726.8	−6947.9	725.8	−7447.9
	3	883.2	−5772.2	883.0	−5932.1	884.6	−6432.2	885.7	−7093.8	885.6	−7593.8
	10	1659.9	−6204.8	1676.6	−6375.1	1678.3	−6880.1	1628.9	−7510.7	1677.6	−8036.1
	50	4278.8	−6638.8	4507.1	−6829.5	5392.0	−7502.2	5542.6	−8273.3	4679.7	−8636.1

The independence of the expected number of oracle calls to the distribution is exhibited as the entries for the same problem instance remain unchanged in all columns labeled EOC. This can also be verified by comparing the corresponding entries of Tables 1 and 2. A cross checking of the columns, where the bisection is reported (the ones labeled EB) with the corresponding columns of Table 1, verifies Theorem 4.3, that is, the bisection phase performs worst when the distribution is uniform. But the results of this table go a step further as they reveal a quantitative nature of this fact. That is, distributions with smaller deviation result in better performance in the bisection phase. This seems to be a more general fact and also holds for all distributions which differ from the uniform in the sense that they possess a "clustering" behavior.

Table 3 records the performance of Algorithm *find_roots* of [10] (that is, equal subdivisions of intervals) applied to instances with Gaussian distributed roots. This table shows an interesting and possibly practical fact, namely, the performance of *find_roots* (which by construction ignores the distribution of roots) to arbitrarily distributed instances is not dramatically worse than the performance of *generalized_find_roots* for small m. Taking into account the difficulty of dividing an interval into equiprobable subintervals, this may suggest that it might be better to ignore the distribution and always subdivide into two equal subintervals. Indeed, in all cases Algorithm *generalized_find_roots* performs better in terms of the EOC when the subdivision is done into two subintervals. This seems to justify choices made by other methods such as interval analysis that divide the interval into two halfs. Additional advantage of this fact is the relative simplicity of the oracle call; an algorithm need not have the exact number of roots in an interval and consequently, the corresponding oracle can be simplified. Taking into account the previous observation and this fact, we conclude that we can also safely ignore the distribution and assume uniformity without much loss in efficiency. In addition the columns reporting the variances also suggest that for $m = 2$ the algorithm presents smaller deviation both in the localization and in the root-finding phase, i.e., it is more predictable. In conclusion, Algorithm *find_roots* when implemented with $m = 2$ identifies clusters of roots almost as rapidly as *generalized_find_roots* and subsequently its subdivisions concentrate where roots are in abundance.

We stress however that this holds when $m = 2$. For larger values of m when required (for example, in a parallel implementation of the algorithms), Algorithm *generalized_find_roots* is clearly better than *find_roots*.

These observations and especially the case of $m = 2$ need theoretical justification which poses an interesting theoretical problem. To show that always dividing by 2 independently of the number of roots for $n \geq 2$ results in a better performance than any other choice of the number of subdivisions. This would in effect show that oracles reporting more than two roots in an interval do not offer more useful information for discovering all the roots.

Acknowledgments. The authors wish to thank the anonymous referees for their constructive comments, suggestions, and valuable criticisms which helped us to improve the paper.

REFERENCES

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[2] 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.

[3] K. M. Brown and W. B. Gearhart, *Deflation techniques for the calculation of further solutions of a nonlinear system*, Numer. Math., 16 (1971), pp. 334–342.

[4] S. Graf, E. Novak, and A. Papageorgiou, *Bisection is not optimal on the average*, Numer. Math., 55 (1989), pp. 481–491.

[5] E. R. Hansen, *A globally convergent interval method for computing and bounding real roots*, BIT, 18 (1978), pp. 415–424.

[6] E. R. Hansen, *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.

[7] E. R. Hansen, *Computing zeros of functions using generalized interval arithmetic*, Interval Comput., 3 (1993), pp. 3–28.

[8] E. R. Hansen and G. W. Walster, *Nonlinear equations and optimization*, Comput. Math. Appl., 25 (1993), pp. 125–145.

[9] 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.

[10] D. J. Kavvadias and M. N. Vrahatis, *Locating and computing all the simple roots and extrema of a function*, SIAM J. Sci. Comput., 17 (1996), pp. 1232–1248.

[11] R. B. Kearfott, *An efficient degree-computation method for a generalized method of bisection*, Numer. Math., 32 (1979), pp. 109–127.

[12] R. B. Kearfott, *Some tests of generalized bisection*, ACM Trans. Math. Software, 13 (1987), pp. 197–220.

[13] R. B. Kearfott, *Interval arithmetic techniques in the computational solution of nonlinear systems of equations: Introduction, examples, and comparisons*, Computational Solution of Nonlinear Systems of Equations, in Lectures in Appl. Math. 26, AMS, Providence, RI, 1990, pp. 337–357.

[14] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[15] R. B. Kearfott and M. Novoa iii, *INTBIS: A portable interval Newton/bisection package*, ACM Trans. Math. Software, 16 (1990), pp. 152–157.

[16] R. Krawczyk and A. Neumaier, *Interval slopes for rational functions and associated and centered forms*, SIAM J. Numer. Anal., 22 (1985), pp. 604–616.

[17] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, UK, 1990.

[18] E. Picard, *Sur le nombre des racines communes à plusieurs équations simultanées*, J. de Math. Pure et Appl. (4^e série), 8 (1892), pp. 5–24.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, New York, 1992.

[20] K. Sikorski, *Bisection is optimal*, Numer. Math., 40 (1982), pp. 111–117.

[21] F. Stenger, *Computing the topological degree of a mapping in* $\mathbb{R}^n$, Numer. Math., 25 (1975), pp. 23–38.

[22] M. Stynes, *An algorithm for numerical calculation of topological degree*, Appl. Anal., 9 (1979), pp. 63–77.

[23] 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.

[24] M. N. Vrahatis, *A short proof and a generalization of Miranda's existence theorem*, Proc. Amer. Math. Soc., 107 (1989), pp. 701–703.

[25] M. N. Vrahatis, T. N. Grapsa, O. Ragos, and F. A. Zafiropoulos, *On the localization and computation of zeros of Bessel functions*, Z. Angew. Math. Mech., 77 (1997), pp. 467–475.

[26] M. N. Vrahatis, O. Ragos, F. A. Zafiropoulos, and T. N. Grapsa, *Locating and computing zeros of airy functions*, Z. Angew. Math. Mech., 76 (1996), pp. 419–422.