SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 93–107

EFFICIENTLY COMPUTING MANY ROOTS OF A FUNCTION*

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

Dedicated to the memory of Árpád Elbert (1939–2001)

Abstract. We present a new bisection based method for counting and computing roots of a function in a given interval. Our method is focused on very large problems, i.e., instances with the number of roots of the order of hundreds. The method draws its power from the fact that the roots are expected to be many, and is able to discover a large percentage of them very efficiently. Its main advantage, apart from its efficiency, is the fact that it requires only the sign of the function at a certain point and not its actual value. Also, its simplicity makes it a suitable preprocessing step for reducing the size of the problem, prior to more robust but also more demanding methods. The algorithm is accompanied by a probabilistic analysis of its behavior, which shows that a simple existence criterion like Bolzano's rule can be a powerful tool in the zerofinding process.

Key words. zerofinding, expected behavior, bisection based methods, counting and computing the roots of a function, very large problems, Riemann's hypothesis, zeta-function, special functions, Elbert's conjecture

AMS subject classifications. 65H05, 65Y20

DOI. 10.1137/S1064827502406531

1. Introduction.

In this paper we present a bisection based method for the problem of counting and computing the simple roots of a single equation:

$$f(x) = 0 \tag{1.1}$$

in a given interval (a, b) where $f: [a, b] \subset \mathbb{R} \to \mathbb{R}$ is continuous. The only computable information required by the proposed method is the algebraic signs of the function values. The method is focused on very large instances of the problem (with roots in the order of hundreds or more).

Pieces of information concerning all of the roots (or a large fraction of them), as well as all the extrema of a function $f: [a, b] \subset \mathbb{R} \to \mathbb{R}$, are of major importance in many different fields in science and technology such as mechanics, physical sciences, statistics, and operations research. For instance, the problem of locating local maxima and minima of a function (or zeros of the derivative of a function) from approximate measurement results is vital for many physical applications. In spectral analysis, chemical species are identified by locating local maxima of the spectra. In radioastronomy, sources of celestial radio emission and their subcomponents are identified by locating local maxima of the measured brightness of the radio sky. Elementary particles are identified by locating local maxima of the experimental curves (for a discussion of the importance of the above mentioned problems for applications, see [28]). The importance of the problem has attracted the attention of many research efforts, and as a result many different approaches to the problem exist. Regarding special functions, for example, Lozier [17] and Lozier and Olver [18] have provided a survey of algorithms and software for the numerical evaluation of special functions.

*Received by the editors April 30, 2002; accepted for publication (in revised form) November 4, 2004; published electronically August 17, 2005.
http://www.siam.org/journals/sisc/27-1/40653.html

†Department of Mathematics, University of Patras, GR–26110 Patras, Greece, and University of Patras Artificial Intelligence Research Center, University of Patras, GR–26110 Patras, Greece (djk@math.upatras.gr, makri@math.upatras.gr, vrahatis@math.upatras.gr).

They have pointed out that the available software for special functions (including the zerofinding approach) exhibits gaps and defects with respect to the needs of modern high-performance computing. In particular, the software regarding the zeros of Bessel functions has a low cumulative score as reported in [17, p. 351], and thus the computation of the zeros of special functions is an area of particular need.

Apart from its practical significance, the problem of computing zeros of a function poses theoretical challenges that have attracted the attention of the scientific community for many years. A characteristic paradigm is the famous *Riemann's hypothesis*, which is one of the main open problems of mathematics (and was included by Hilbert as Problem No. 8 in his famous 1900 list of 23 problems). According to this hypothesis, all complex roots $s_1 = \Re(s_1) + i\,\Im(s_1), s_2 = \Re(s_2) + i\,\Im(s_2), \ldots$, of Riemann's ζ-function (zeta-function)

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

(i.e., the values for which $\zeta(s_k) = 0$) are located on the straight line $\Re(s_k) = 1/2$ in the complex plane (except for the known zeros which are negative integers). It is known that the imaginary parts of the roots of Riemann's ζ-function are uniformly distributed. Riemann's ζ-function has proved to be of fundamental importance not only in the theory of prime numbers, but also in the higher theory of the Gamma-function and allied functions. Numerical computation of the roots of the ζ-function in a specific interval may gather evidence that hopefully will help in resolving the problem.

In addition, the task of massive calculation of zeros of functions in one variable also emerges naturally in recently posed problems. Such a problem is *Elbert's conjecture*. More specifically, in [3, p. 75] Elbert pointed out that the density property of the zeros of Bessel functions plays an important role, and better insight into the distribution of these zeros is required (see also [11, 12], where Joó dealt with oscillation of circular membranes). Elbert considered the set

$$\mathcal{S} = \bigcup_{n,k=1}^{\infty} \{j_{nk}\} = \bigcup_{j=1}^{\infty} \{x_j\},$$

where j_{nk} is the kth positive zero of the Bessel function of first kind, $J_n(x)$, and $x_1 < x_2 < \cdots$, and he conjectured that [3]

$$\limsup_{j\to\infty} x_j(x_{j+1} - x_j) < \infty.$$

For results on the massive calculation and more information on the values $x_j(x_{j+1} - x_j)$, see [23].

The proposed method takes advantage of the abundance of roots, in an attempt to very inexpensively (in terms of computing resources) locate a large number of roots, thus reducing the size of the problem. The algorithm almost "blindly" searches for roots by employing only Bolzano's existence criterion (see next section). It turns out, however, that even this simple rule is sufficient to guide the algorithm in discovering a large proportion of the total set of roots. Thus with very few sign determinations, the problem can be considerably reduced. But this is already more than could be expected from such a simple method: While new roots are being discovered, the cost

gets increasingly high to a point where it becomes unprofitable to continue. The algorithm stops when a predetermined fraction of the roots has been discovered. This requires a method of estimating the number of roots as the algorithm proceeds. After the termination of the algorithm the unsearched parts of the interval must be searched by a different, more demanding, method. The main advantage of the proposed method is its simplicity, which follows precisely from the expectation that with high probability even a simple search for a root will prove successful due to the abundance of roots.

In this paper we also study analytically the expected behavior of the method and give theoretical justification of its good performance.

We mention that there has been a surge of interest concerning the expected behavior of numerical algorithms [6, 7, 8, 20, 24, 27] and [13, 14]. The traditional approach in evaluating a numerical method usually involves a number of experiments on a number of inputs, either of individual interest or randomly constructed by altering certain parameters of the problem. Very few numerical methods exist that are accompanied by a robust analysis of their expected behavior.

A by-product of the algorithm is an estimation of the number of roots in an interval. This problem is interesting on its own, either to a priori evaluate the size of a problem or, as in this case, to establish a stopping criterion for the method.

Other approaches that have been used, aiming 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, 9, 15, 19]). The corresponding existence tool of these methods is the availability of the range of the function in a given interval, which can be implemented using interval arithmetic, although range overestimation may occur, and hence efficiency problems must be resolved. This tool will, with mathematical rigor, give either a "no" or an "unknown" answer. The former case proceeds by subdividing the interval into two halves and employing additional criteria. The way the evaluation of functions is encoded influences the answer, which is usually pessimistic (i.e., "unknown"). In the vicinity of a root, interval Newton methods (see, e.g., [15, 19]) may, however, with slightly more computational effort, give an unambiguous "yes" answer.

In the next section we give some background material on the bisection method, as was modified in [29, 30]. In section 3 we briefly discuss the main steps of the algorithm to present a (central to our analysis) probabilistic result which follows in section 4. Then in section 5 we give a method for estimating the total number of roots. A more detailed description of the method, which also refers to the theoretical analysis, is presented in section 6. We end in section 7 by presenting some conclusions and future work.

2. Background material. 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:

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

where sgn is the well-known three-valued sign function. This criterion is known as Bolzano's existence criterion. (For a generalization of this criterion to higher dimensions, see [31].) 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.

The simplicity of this criterion is what makes it attractive even though it has the above disadvantage. Thus it will be our main tool in what follows. More elaborate relations can give more information on the existence of roots. For example, interval

analysis uses the range of the function to decide whether a root exists (see, e.g., [15, 19]). Another approach is given in [16]. An even more complicated 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 [10, 21]. This oracle was used in [14] as part of the first phase of an algorithm for the isolation of all of the simple roots of a function $f(x)$ in an interval (a, b) and returns the number of roots $\mathcal{N}^r$ using the formula

$$\mathcal{N}^r = -\frac{\gamma}{\pi}\int_a^b \frac{f(x)f''(x) - f'^2(x)}{f^2(x) + \gamma^2 f'^2(x)}\,dx + \frac{1}{\pi}\arctan\left(\frac{\gamma\,[f(a)f'(b) - f(b)f'(a)]}{f(a)f(b) + \gamma^2 f'(a)f'(b)}\right), \tag{2.1}$$

where γ is an arbitrary small real positive constant, i.e., $\gamma \simeq 1$. (For a variation of the CPU time for the computation of $\mathcal{N}^r$ versus γ, see [36].) It was explicitly shown by Picard [21, 22] that relation (2.1) is independent of the value of γ.

The algorithm of [14] uses bisection in its second phase to compute the roots. Specifically, it uses the following simplified version described in [29]:

$$x_{i+1} = x_i + c\,\mathrm{sgn}f(x_i)/2^{i+1}, \quad i = 0, 1, \ldots, \tag{2.2}$$

where $c = \mathrm{sgn}f(a)\,(b - a)$. The sequence (2.2) converges to a root $r \in (a, b)$ if for some x_i,

$$\mathrm{sgn}f(x_0)\,\mathrm{sgn}f(x_i) = -1, \quad \text{for} \quad i = 1, 2, \ldots.$$

Furthermore, the number of iterations ν which are required to obtain an approximate root r^* such that $|r - r^*| \leqslant \varepsilon$ for some $\varepsilon \in (0, 1)$ is given by

$$\nu = \left\lceil \log(b - a)\,\varepsilon^{-1} \right\rceil, \tag{2.3}$$

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 (2.2) we can also use the following one:

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

where $c = \mathrm{sgn}f(b)\,(b - a)$.

The reason for choosing the bisection method is that it always converges within the given interval (a, b) and is a globally convergent method. Moreover it has a great advantage since it is worst-case optimal; i.e., it possesses asymptotically the best possible rate of convergence in the worst case [25, 26]. This means that it is guaranteed to converge within the predefined number of iterations, and, moreover, no other method has this property. Therefore, using relation (2.3) 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, the bisection method requires only the algebraic signs of the function values to be computed, as is evident from (2.2) or (2.4); 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 [35, 37] that the sign stabilizes after a relatively small number of terms of the series and the calculations can be sped up considerably.

3. A high level description of the method. Here is an informal description of the algorithm. For the detailed algorithm, see subsection 6.1. The algorithm gets as input the fraction of the roots that are required to be computed and begins by

subdividing the interval into two equal subintervals. Alternatively, it may get as input a predetermined "budget" of function evaluations. The main body of the algorithm consists of three steps.

Step 1. A number of subintervals stored from previous steps, enter Step 1. These subintervals resulted from dividing the original interval, and at the first iteration these are just the two halves mentioned above. We determine the sign of the function at the endpoints of these subintervals. Depending on the number of simple roots in each subinterval, its endpoints will have the same signs (in the case of even roots) or opposite signs (in the case of odd roots). We therefore have certainty on the existence of at least one simple root in an interval with opposite signs, so we proceed in discovering a root in those intervals using the bisection method.

Step 2. Note that at this point our interval has been divided into subintervals whose endpoints are of the same sign. These subintervals are kept for further examination if our stopping criterion has not been fulfilled yet. We propose two variants of a stopping criterion. The first (which we consider to be more interesting and suggest as the main variant of the algorithm) requires that a fraction of the total number of roots must be discovered before terminating the algorithm. The second terminates the algorithm if a predefined budget of function evaluations has expired.

Step 3. If the stopping criterion has not been fulfilled, we subdivide the set of subintervals with maximum length (among all subintervals that have been stored) into two halves and go back to Step 1. Otherwise we output the discovered roots and halt.

Observe that the algorithm uses the function only to compute its sign at specific points during its execution. Thus, if the sign is available by other means, apart from a direct calculation (and this mainly refers to an inexact calculation using some kind of approximation of the function value), the algorithm can still run without any problem [4, 5, 35, 37]. As a result, the actual problem that the algorithm solves is that of discovering N hidden points in the interval, using only an *oracle* that answers queries of the following form: "Is the number of points in a specific subinterval (a, b) odd or even?"

4. The distribution of odd subintervals. In this section we find the probability of having k specific subintervals containing an odd number of roots. The probability space on which we make our calculations and the whole framework can be found in [14]. We briefly mention here that we view each root as a random point in the interval $(0, 1)$. That is, we assume without loss of generality that the given interval is normalized, i.e., its length is 1. This assumption is followed throughout this paper. We also make the assumption that the roots are *randomly* and *uniformly* distributed; i.e., intervals of equal length have equal probability of containing a root. Consequently, the length ℓ of a subinterval also gives the probability of a randomly chosen point to lie within this subinterval.

Consider the following situation: N points are chosen randomly and uniformly in the interval $(0, 1)$. Assume that we have chosen m nonoverlapping subintervals all of the same length ℓ. We view the remaining part of the interval (which may not be connected) as the $(m + 1)$-st subinterval.

Let us denote by E the event that each one of k specific subintervals contains an odd number of roots and the remaining $m - k$ contain an even number of roots. The following theorem gives the exact probability of the event E.

THEOREM 4.1. *Assume that in the interval* $(0, 1)$*,* N *points have been selected randomly and uniformly. Assume also that* m *nonoverlapping subintervals of the same*

length ℓ have been chosen. Then the probability that k specific subintervals among the m contain an odd number of points is given by the formula

$$\mathrm{P}(E) = \frac{1}{2^m} \sum_{S} \binom{N}{S} \ell^S (1 - m\ell)^{N-S} \sum_{j=0}^{m} (m - 2j)^S \sum_{i=0}^{j} (-1)^i \binom{k}{i} \binom{m-k}{j-i}. \tag{4.1}$$

Proof. We associate with the ith subinterval, $i = 1, 2, \ldots, m$, a random variable X_i, $i = 1, 2, \ldots, m$, that takes the value 1 if the number of points in the subinterval is odd and 0 if it is even. These variables are identically distributed since we have assumed that the roots are uniformly distributed and the subintervals have equal length. The probability of the event E is given by

$$\mathrm{P}(E) = \mathrm{P}(X_{i_1} = 1, \ldots, X_{i_k} = 1, X_{i_{k+1}} = 0, \ldots, X_{i_m} = 0),$$

where $i_1, i_2, \ldots, i_m$ is a permutation of the set $\{1, 2, \ldots, m\}$. Notice that $\mathrm{P}(E)$ is independent of the specific permutation $i_1, i_2, \ldots, i_m$ since a point has the same probability equal to ℓ of being in each of the m subintervals.

The arrangement of the N points in the $m + 1$ subintervals can therefore be seen as a repetition N times of an experiment with $m + 1$ possible outcomes, the first m of which each have probability ℓ, and the last $(1 - m\ell)$. Let S, $0 \leqslant S \leqslant N$ be the total number of points in the m subintervals. The number of ways the subintervals $i_1, i_2, \ldots, i_m$ will contain, respectively, $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_m}$ points is given by the multinomial coefficient $\binom{S}{\nu_{i_1}, \ldots, \nu_{i_m}}$. Hence we get for $\mathrm{P}(E)$

$$\mathrm{P}(E) = \sum_{S} \binom{N}{S} \sum \binom{S}{\nu_{i_1}, \ldots, \nu_{i_m}} \ell^{\nu_{i_1} + \cdots + \nu_{i_m}} (1 - m\ell)^{N - (\nu_{i_1} + \cdots + \nu_{i_m})}, \tag{4.2}$$

where the inner summation is over all nonnegative integers $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_m}$ satisfying the conditions $\nu_{i_1} + \nu_{i_2} + \cdots + \nu_{i_m} = S$ with $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_k}$ being odd numbers and $\nu_{i_{k+1}}, \nu_{i_{k+2}}, \ldots, \nu_{i_m}$ being even. The above equation therefore becomes

$$\mathrm{P}(E) = \sum_{S} \binom{N}{S} \ell^S (1 - m\ell)^{N-S} \sum \binom{S}{\nu_{i_1}, \ldots, \nu_{i_m}}. \tag{4.3}$$

The proof now rests in finding a more easily computable form for the sum of multinomial coefficients, since the number of different tuples of $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_m}$ that fulfill the requirements is very large for any reasonable problem size, not to mention for computing large factorials.

Notice that the sum of multinomial coefficients (which we denote by $C(S, m, k)$) gives the number of permutations of S out of m distinct objects with repetitions such that k specific objects are each selected an odd number of times and the remaining $m - k$ are selected an even number of times.

Let us denote by $B(u, v)$ the number of v-permutations of u objects with repetition where each object is selected an odd number of times. In [2, p. 227] $B(u, v)$ is given

by

$$B(u,v) = \begin{cases} 2^{v-u+1} \displaystyle\sum_{j=0}^{\frac{u}{2}-1} (-1)^j \binom{u}{j} \left(\frac{u}{2}-j\right)^v, & u \text{ even, } v \text{ even,} \\ 0, & u \text{ even, } v \text{ odd,} \\ 2^{v-u+1} \displaystyle\sum_{j=0}^{\frac{u-1}{2}} (-1)^j \binom{u}{j} \left(\frac{u}{2}-j\right)^v, & u \text{ odd, } v \text{ odd,} \\ 0, & u \text{ odd, } v \text{ even.} \end{cases} \tag{4.4}$$

Let us also denote by $A(u,v)$ the number of v-permutations of u objects with repetition where each object is selected an even number of times. In the same reference, $A(u,v)$ is given by

$$A(u,v) = \begin{cases} 2^{-u} \displaystyle\sum_{j=0}^{u} \binom{u}{j} (u-2j)^v, & v \text{ even,} \\ 0, & v \text{ odd.} \end{cases} \tag{4.5}$$

So if we assume that ν points in total lie in the k subintervals which are expected to have an odd number of points, the remaining $S-\nu$ lie in the remaining $m-k$ subintervals. We may therefore express $C(S,m,k)$ in terms of $A(u,v)$ and $B(u,v)$ as follows:

$$C(S,m,k) = \sum_{\nu=k}^{S} \binom{S}{\nu} B(k,\nu) A(m-k, S-\nu). \tag{4.6}$$

This sum can now be transformed using (4.4) and (4.5), resulting in

$$C(S,m,k) = \frac{1}{2^m} \sum_{j=0}^{m} (m-2j)^S \sum_{i=0}^{j} (-1)^i \binom{k}{i} \binom{m-k}{j-i}. \tag{4.7}$$

Substituting the above in (4.3) we get (4.1). □

When the initial interval is partitioned into m equal subintervals, i.e., when $m\ell = 1$, then all N roots must necessarily lie in the m subintervals. Let us denote E' the event that k specific subintervals have an odd number of roots under this assumption. We have the following corollary.

COROLLARY 4.1. *The probability of the event E' is given by*

$$\mathrm{P}(E') = \frac{1}{2^m m^N} \sum_{j=0}^{m} (m-2j)^N \sum_{i=0}^{j} (-1)^i \binom{k}{i} \binom{m-k}{j-i}. \tag{4.8}$$

Proof. In this situation there are only m possible outcomes of the experiment, all of the same probability ℓ. The analysis above that resulted in (4.1) now gives the above formula. □

5. Estimating the total number of roots. Our stopping criterion in the main variant of the algorithm requires the knowledge of the total number of roots in the interval. For this purpose, the algorithm, in parallel with its main task of computing roots, also estimates the total number of roots. The estimation of the total number of

roots is revised after each iteration, when new roots have been discovered. Hence, at the beginning of the algorithm there is only a rough estimation of the number of roots, but as we proceed the additional information allows us to be more accurate in our prediction. For our estimation method we shall need the following two propositions.

PROPOSITION 5.1. *Assume that the total number of roots in the interval is N. Then the probability p_{odd} that a subinterval of length ℓ contains an odd number of roots is given by*

$$p_{\text{odd}} = \frac{1-(1-2\ell)^N}{2}. \tag{5.1}$$

Proof. Let p_{even} be the probability that a subinterval of length ℓ contains an even number of roots. Then clearly $p_{\text{odd}} = 1 - p_{\text{even}}$. Recall that ℓ is the probability of having a specific root in an interval of length ℓ. Now

$$p_{\text{even}} = (1-\ell)^N + \binom{N}{2}\ell^2(1-\ell)^{N-2} + \binom{N}{4}\ell^4(1-\ell)^{N-4} + \cdots.$$

Observe that the right-hand side equals $\frac{1}{2}\left[(x+y)^N + (x-y)^N\right] = \frac{1}{2}\left[1+(1-2\ell)^N\right]$ for $x = 1-\ell$ and $y = \ell$. □

Notice that for a number of roots N on the order of 20 or so, p_{odd} is very close to $1/2$ for large enough ℓ. Therefore, roughly half of the intervals have at least one root. This is natural, since when the number of roots is large, the intervals with odd roots and the intervals with even roots are expected to be about the same. In such a case, after computing the roots, it is worthwhile continuing by subdividing the largest intervals. When, however, we have discovered a number of roots close to the total (and thus the remaining roots are few), the number of intervals with no roots begins to increase, and thus the required sign evaluations per root also increase.

To quantify the above, consider the random variables X_i, $i = 1, \ldots, m$, introduced in the previous section. These variables are not independent since, for example, if $m = N + 1$ and we know that $X_i = 1$ for $i = 1, 2, \ldots, m-1$, the event $X_m = 0$ has probability 1; i.e., the probability of an X_i taking a certain value is influenced by the values of the other variables. This dependency is, however, very weak for large values of N in the sense that the joint probability distribution function of the X_i's is very close, yet not equal, to the product of the marginal distribution functions.

If the number of roots in any of the subintervals were independent from the number of roots in the other subintervals, then the probability of the event E (see the previous section) would be given by the formula $p_2 = (p_{\text{odd}})^k(1-p_{\text{odd}})^{m-k}$. This follows from the observation that the probability of having odd roots is p_{odd} for all subintervals of length ℓ, which along with the assumption of independence gives the above. What we show next is that the exact probability p_1 of E as given by (4.1) is actually very close to p_2. We demonstrate this by comparing p_1 and p_2 numerically.

In Table 5.1 we have included a small fraction of extensive numerical comparisons of p_1 and p_2 for several values of the parameters N, m, k, and ℓ. To reduce the size of the table, we present here results for ℓ fixed to a typical value of $m/0.8$. The results show that p_1 and p_2 are close enough to be considered equal for the purpose of estimating the number of roots N.

It is therefore a satisfactory approximation to consider the X_i's as independent Bernoulli random variables with probability given by (5.1). The following is a straightforward estimation of this probability.

TABLE 5.1
Numerical comparison of p_1 and p_2 for several values of the parameters N, m, and k.

N	m	k	p_1	p_2
100	20	2	0.952×10^{-6}	0.953×10^{-6}
100	20	15	0.959×10^{-6}	0.959×10^{-6}
100	50	5	0.367×10^{-14}	0.404×10^{-14}
100	50	40	0.246×10^{-15}	0.268×10^{-15}
100	120	12	0.101×10^{-27}	0.153×10^{-26}
100	120	70	0.333×10^{-41}	0.514×10^{-40}
500	100	10	0.808×10^{-30}	0.808×10^{-30}
500	600	60	0.816×10^{-134}	0.161×10^{-133}
1000	200	40	0.152×10^{-64}	0.454×10^{-64}
1000	200	160	0.596×10^{-60}	0.596×10^{-60}
1000	500	100	0.213×10^{-145}	0.390×10^{-145}
1000	500	400	0.393×10^{-156}	0.104×10^{-155}

PROPOSITION 5.2. *Assume that in a family of m nonoverlapping subintervals all of the same length ℓ, k subintervals have opposite signs at their endpoints. Then, with probability at least $(1-\alpha)$, for any α between 0 and 1, the probability p_{odd} that any subinterval of length ℓ has an odd number of roots is between*

$$p_{\text{lower}} \leqslant p_{\text{odd}} \leqslant p_{\text{upper}},$$

where

$$p_{\text{lower}} = \frac{k - z_{\alpha/2}\sqrt{\frac{k(m-k)}{m}}}{m},$$

and

$$p_{\text{upper}} = \frac{k + z_{\alpha/2}\sqrt{\frac{k(m-k)}{m}}}{m},$$

where $z_{\alpha/2}$ is a constant which can be determined from

$$\text{Prob}(-z_{\alpha/2} \leqslant Z \leqslant z_{\alpha/2}) = 1-\alpha,$$

where Z is a random variable having the standard normal distribution.

Proof. Consider the m subintervals. Let X_i be a random variable assuming value 1 if the ith subinterval has opposite signs and 0 otherwise, $i = 1, \ldots, m$. By the discussion above, the $X_i, i = 1, \ldots, m$, are considered to be independent Bernoulli random variables $B(1, p_{\text{odd}})$. A confidence interval for p_{odd} with approximate confidence coefficient $(1-\alpha)$ is given by

$$\frac{k}{m} - z_{\alpha/2}\sqrt{\frac{\frac{k}{m}\left(1-\frac{k}{m}\right)}{m}} \leqslant p_{\text{odd}} \leqslant \frac{k}{m} + z_{\alpha/2}\sqrt{\frac{\frac{k}{m}\left(1-\frac{k}{m}\right)}{m}},$$

which proves the proposition. ∎

For example, when $\alpha = 0.05$, then $z_{\alpha/2} = 1.96$. We use the above confidence interval to estimate p_{odd} and then use (5.1) to estimate N. This is done by computing

two values for N, calling them N_{lower} and N_{upper}. The first is computed by solving the equation (with respect to N)

$$p_{\text{lower}} = \frac{1-(1-2\ell)^N}{2}, \tag{5.2}$$

and the second is computed by solving the equation

$$p_{\text{upper}} = \frac{1-(1-2\ell)^N}{2}. \tag{5.3}$$

Notice that in some cases N_{upper} and even N_{lower} can be infinite. This will happen when insufficient data are available for determining exact values for the confidence interval. But for our purposes this is immaterial: in either of these cases we continue subdividing the intervals.

When after some iterations, both ends of the confidence interval are well defined, we choose some preferable value for N. A good choice seems to be $(N_{\text{lower}}+N_{\text{upper}})/2$. Or, we may decide to be on the safe side and choose N to be N_{upper}. Whichever we choose, we use it to decide whether it is time to terminate.

6. The method and its theoretical analysis. In this section we study the expected behavior of the algorithm as a function of the problem size N and the required fraction of roots λ. Our assumption is that, using the method described in the previous section, we know the total number of roots N, even though this can only be determined approximately within the confidence interval. Knowing N, the algorithm will run until a fraction λ of the roots has been discovered, i.e., λN roots. Clearly λ can vary from values close to 0, resulting in a quick termination of the algorithm, to values close to 1, which increase the cost of discovering new roots to forbiddingly high levels, after a large percentage has been discovered. Such values of λ will render the algorithm very costly and should not be used. We next give the expected cost of the algorithm for specific λ. To this end, we first give a technical definition that facilitates our analysis.

DEFINITION 6.1. *We call* iteration number *of the algorithm the integer i such that the length ℓ of the subintervals with an even number of roots that have maximum length, is such that $\ell = 2^{-i}$.*

The reason for this definition is that at Step 1 of the algorithm, after dividing the intervals, we would typically expect some subintervals to have an even number and some to have an odd number of roots. If at some stage of execution of the algorithm all previous executions of Step 1 gave subintervals of both types or only even ones, then the iteration number i would coincide with what one would normally expect as iteration, since one execution of the three steps results in dividing the length of the largest subintervals by 2. But there is also the possibility that the intervals entering Step 1 are all of odd roots. In this case all subintervals would be further divided by bisection in Step 1 in the same "ordinary" iteration. We therefore give the above definition to relate the length of the largest even subintervals with the iteration number. Notice that the event of having all subintervals with an odd number of roots has small probability (actually it could only happen with some meaningful probability in the first few iterations). Therefore the iteration i as described above is an integer very close to the "ordinary" iteration. Note also that if the algorithm is in iteration i, then the effect on the original interval is the same as directly dividing it into 2^i equal subintervals and executing bisection in those subintervals that have an odd number of roots.

The next theorem gives the expected number of odd subintervals and hence the expected number of discovered roots in iteration i.

THEOREM 6.1. *Assume that the interval* $(0,1)$ *has been partitioned into* m *equal subintervals. Then the expected number of those that contain an odd number of roots is*

$$\mathrm{E}_d = \frac{m^N - (m-2)^N}{2m^{N-1}}. \tag{6.1}$$

Proof. The probability of getting, after the division into m subintervals, k odd subintervals is given by $\binom{m}{k}\mathrm{P}(E')$. E' is the event where, after the division into m subintervals, k *specific* intervals have an odd number of roots and the remaining $m-k$ have an even number. This event was formally defined in section 4 where its probability also was derived and is given by (4.8). Hence we have

$$\begin{aligned}
\mathrm{E}_d &= \sum_{k=0}^{m} k\binom{m}{k}\mathrm{P}(E') \\
&= \frac{1}{2^m m^N}\sum_{k=0}^{m} k\binom{m}{k}\sum_{j=0}^{m}(m-2j)^N\sum_{i=0}^{j}(-1)^i\binom{k}{i}\binom{m-k}{j-i} \\
&= \frac{1}{2^m m^N}\sum_{j=0}^{m}(m-2j)^N\binom{m}{j}\sum_{i=0}^{j}(-1)^i\binom{j}{i}\sum_{k=0}^{m} k\binom{m-j}{k-i}.
\end{aligned}$$

Now the rightmost sum is easily seen to be $(m-j)2^{m-j-1} + i2^{m-j}$. By substituting and after some algebraic manipulations, the theorem follows. ∎

Assume now that a fraction λ of the roots is required. In the ith iteration we have $m = 2^i$ and by substituting we get that the iteration where this is achieved, on average, is the solution rounded above, for i, of the equation

$$\lambda N = \frac{2^{iN} - (2^i - 2)^N}{2^{i(N-1)+1}}. \tag{6.2}$$

Table 6.1 gives the solution of (6.2) for various values of N and λ.

TABLE 6.1
Iteration and expected work for achieving a fraction of roots λ.

		N = 100		500		1000		5000	
		i	w	i	w	i	w	i	w
λ	0.5	7	776	10	3508	11	6515	13	25522
	0.7	8	1093	11	5175	12	9650	14	37146
	0.9	10	1919	13	11313	14	21724	16	83230
	0.95	11	2898	14	19203	15	37454	17	144998

As for the expected work for achieving the fraction λ, notice that at iteration i, where d odd subintervals have been discovered, $(2^i + 1)$ sign determinations are required to split the interval into 2^i subintervals, and if d of those are odd, then bisection will be applied to each of them, requiring $\log(\ell/\varepsilon)$ additional sign determinations where ℓ is the length of the subinterval. But $\ell = 2^{-i}$, and i is the solution of (6.2). Hence, we have the following corollary.

COROLLARY 6.1. *The expected work for discovering a fraction λ of the roots is given by*

$$w = 2^i + \lambda N \log \frac{\ell}{\varepsilon} + 1 \equiv 2^i - \lambda N(i + \log \varepsilon) + 1, \tag{6.3}$$

where i is the solution of (6.2).

Table 6.1 also gives the value of the expected work for various values of N and λ.

Another interesting aspect of the behavior of the algorithm is the increase of *the work per root* as a function of the iteration. Since an exact calculation of this seems to be hard, we approximate it as follows.

Let E_{d_i} be the expected number of discovered roots by the end of iteration i. Then, the expected number of discovered roots in iteration $i+1$ is $\mathrm{E}_{d_{i+1}} - \mathrm{E}_{d_i}$. Also, using Corollary 6.1, the expected work in iteration $i+1$ is $2^i - (i + \log \varepsilon)(\mathrm{E}_{d_{i+1}} - \mathrm{E}_{d_i}) - \mathrm{E}_{d_{i+1}}$. We approximate the expected work per root in iteration $i+1$ by the fraction of the expected work in iteration $i+1$ divided by the expected number of discovered roots in the same iteration. We call this fraction E^*_{i+1}. By the above discussion this is given by

$$E^*_{i+1} = \frac{2^i - \mathrm{E}_{d_{i+1}}}{\mathrm{E}_{d_{i+1}} - \mathrm{E}_{d_i}} - i - \log \varepsilon. \tag{6.4}$$

In Figure 6.1 we plot E^*_i for $N = 1000$ and $\varepsilon = 10^{-6}$ as a function of i. We have chosen to present the plot as a continuous curve, even though i is an integer, to better demonstrate its shape. It is interesting to note that the expected cost per root (as approximated by E^*) first decreases slightly and, after reaching a minimum, subsequently increases rapidly. The first part of the plot is explained since in the first iterations, discovering odd intervals is easy (about half of the intervals are odd), but computing the roots in those intervals costs more to the bisection (computing a root in an interval requires one more sign determination than in an interval of half the length). The subsequent rapid increment is caused by the fact that after some iteration, the odd intervals are becoming rare and increasingly more function evaluations are spent dividing an even interval and getting two even subintervals. After that point the behavior of the algorithm deteriorates rapidly and it soon becomes inefficient. This is the point to stop execution.

6.1. A detailed description of the proposed algorithm. In this section we give a detailed description of the proposed algorithm based on the previous results and analysis. The algorithm takes as input the desired fraction of the roots, λ.

1. Divide the interval into two equal subintervals. Set $i = 1$ and $m = 2$.
2. Let A be the set of subintervals with opposite signs at their endpoints and let k be its cardinality, i.e., $k = |A|$. Let B be the set of subintervals with the same signs at their endpoints.
3. Find one root in each interval in A using bisection (2.2) or (2.4). As bisection proceeds, it leaves out intervals with an even number of roots. Add those into set B.
4. Estimate the total number of roots using relations (5.2) and (5.3), where p_{lower} and p_{upper} are given in Proposition 5.2. In these equations take $\ell = 2^{-i}$, where i is the number of the current iteration, and use the parameters $m = 2^i$ and k defined above.
5. If both N_{lower} and N_{upper} are finite, check whether $d \geqslant \lambda N$. $N = (N_{\text{lower}} + N_{\text{upper}})/2$, and d is the number of discovered roots up to this point.

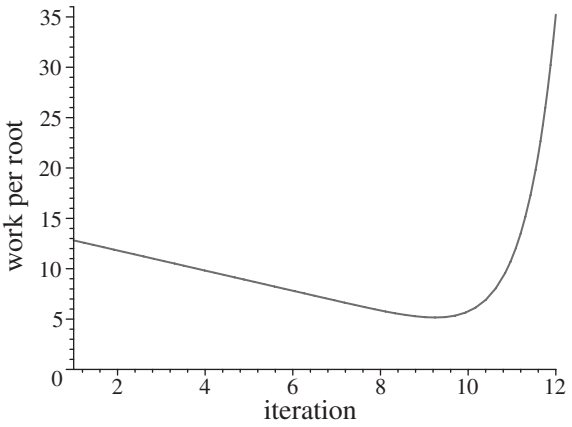

FIG. 6.1. *Expected work per root vs. iteration ($N = 1000$).*

6. If the criterion is not fulfilled or at least one of the N is infinite, then replace in B each subinterval *of largest size* by its two halves. Set $i = i + 1$ and $m = 2^i$ and go to step 2.
7. If the stopping criterion of step 5 is fulfilled, then output the roots and terminate.

A useful observation that helps in improving the efficiency of the code (though it is not our goal to give implementation details here) is that set B is best implemented as a *heap*, i.e., a data structure that allows easy insertions and removals of its items and, what is most important, easy location of the largest element (in constant time). This is needed in steps 3 (insertion) and 6 (find max and removal). Actually, B is best viewed as a set consisting of subsets, where each subset contains subintervals of the same size. These sets are kept sorted by the heap structure, to minimize access time to the largest ones, as required in step 6.

The subdivision into intervals which are fractions that are a power of two is justified as follows: Recall that after we identify an interval with opposite signs at its endpoints, we proceed in discovering a root using bisection (this is step 3 above). But this has the side effect that these intervals will be subdivided further in the process of discovering the root. Now if we choose the initial subintervals to be a power of two, all of these subdivisions that emerge from bisection can be used in the future should the algorithm proceed to subsequent iterations, and thus some sign determinations can be saved. Moreover, this simplifies our arithmetic since each subinterval may be represented not by two real numbers but by its left endpoint and its size, which, in turn, may be represented by two integers, the exponents of actual numbers.

7. Conclusion and further research. As a conclusion, we have addressed the problem of computing the roots of a function when the number of roots is very large. This is a formidable problem with many applications in various fields of science. It seems possible, however, to attack the problem precisely by taking advantage of its size. To this end we have presented an algorithm that effectively discovers roots using an almost "blind" search up to a point where the original size of the problem has been greatly reduced. Then, more robust and expensive methods can be used to completely solve the problem. We have also given theoretical justification of its good performance, based on a probabilistic framework. The main advantages of the

proposed methodology are its simplicity, which results in fairly simple programming, and its efficiency, which increases with the problem size.

There are several possible directions to which this work may be extended. One very natural one is to consider an arbitrary distribution of roots, along the lines of [13]. An interesting problem in this case is to transform the method of estimating the number of roots that we presented here, in a way that takes into account the given distribution.

Moreover, there is no reason to limit the search space to a single dimension. With appropriate assumptions and, most of all, an appropriate existence criterion similar to Bolzano's, for instance, the Poincaré–Miranda hypercubes, Sperner simplices, or the characteristic polyhedra [16, 29, 30, 31, 32, 33, 34], it might be possible to extend the method to higher dimensions.

Finally, the most interesting extension is, of course, to apply the algorithm on a natural, real problem like the ones mentioned in the introduction.

Acknowledgment. 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] Ch. Charalambides, *Enumerative Combinatorics*, CRC Press Ser. Discrete Math. Appl., Chapman & Hall/CRC, Boca Raton, FL, 2002.

[3] Á. Elbert, *Some recent results on the zeros of Bessel functions and orthogonal polynomials*, J. Comput. Appl. Math., 133 (2001), pp. 65–83.

[4] I.Z. Emiris, B. Mourrain, and M.N. Vrahatis, *Sign methods for counting and computing real roots of algebraic systems*, Rapport de recherche 3669, INRIA (Institut National de Recherche en Informatique et en Automatique), Sophia Antipolis, France, 1999.

[5] I.Z. Emiris, B. Mourrain, and M.N. Vrahatis, *Sign methods for enumerating solutions of nonlinear algebraic systems*, in Hellenic European Research on Computer Mathematics and its Applications (HERCMA 2001), Vol. 2, E.A. Lipitakis, ed., L.E.A. Press, Athens, 2001, pp. 469–473.

[6] S. Graf, R.D. Mauldin, and S.C. Williams, *Random homeomorphisms*, Adv. Math., 60 (1986), pp. 239–359.

[7] S. Graf and E. Novak, *The average error of quadrature formulas for functions of bounded variation*, Rocky Mountain J. Math., 20 (1990), pp. 107–716.

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

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

[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] I. Joó, *Exact Controllability and Oscillation Properties of Circular Membranes*, Dissertation for the title doctor of science, Budapest, Hungary, 1992.

[12] I. Joó, *On the control of a circular membrane* I, Acta Math. Hungar., 61 (1993), pp. 303–325.

[13] D.J. Kavvadias, F.S. Makri, and M.N. Vrahatis, *Locating and computing arbitrarily distributed zeros*, SIAM J. Sci. Comput., 21 (1999), pp. 954–969.

[14] 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.

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

[16] B. Mourrain, M.N. Vrahatis, and J.C. Yakoubsohn, *On the complexity of isolating real roots and computing with certainty the topological degree*, J. Complexity, 18 (2002), pp. 612–640.

[17] D.W. Lozier, *Software needs in special functions*, J. Comput. Appl. Math., 66 (1996), pp. 345–358.

[18] D.W. Lozier and F.W.J. Olver, *Numerical evaluation of special functions*, in Mathematics of Computation 1943–1993: A Half-Century of Computational Mathematics, Proc. Sympos. Appl. Math. 48, W. Gautschi, ed., AMS, Providence, RI, 1994, pp. 79–125.

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

[20] E. Novak, K. Ritter, and H. Woźniakowski, *Average-case optimality of a hybrid secant-bisection method*, Math. Comp, 64 (1995), pp. 1517–1539.

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

[22] E. Picard, *Traité d'analyse*, 3rd ed., Gauthier–Villars, Paris, 1922, chap. 4.7.

[23] V.P. Plagianakos, N.K. Nousis, and M.N. Vrahatis, *Locating and computing in parallel all the simple roots of special functions using PVM*, J. Comput. Appl. Math., 133 (2001), pp. 545–554.

[24] K. Ritter, *Average Case Analysis of Numerical Problems*, Lecture Notes in Math. 1733, Springer-Verlag, Berlin, 2000.

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

[26] K. Sikorski, *Optimal Solution of Nonlinear Equations*, Oxford University Press, New York, 2001.

[27] J.F. Traub, G.W. Wasilkowski, and H. Woźniakowski, *Information–based complexity*, Academic Press, New York, 1988.

[28] K. Villaverde and V. Kreinovich, *A linear-time algorithm that locates local extrema of a function of one variable from interval measurements results*, Interval Computations, 4 (1993), pp. 176–194.

[29] 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.

[30] M.N. Vrahatis, *CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations*, ACM Trans. Math. Software, 14 (1988), pp. 330–336.

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

[32] M.N. Vrahatis, *An efficient method for locating and computing periodic orbits of nonlinear mappings*, J. Comput. Phys., 119 (1995), pp. 105–119.

[33] M.N. Vrahatis, *A generalized bisection method for large and imprecise problems*, in Scientific Computing and Validated Numerics, G. Alefeld, A. Frommer, and B. Lang, eds., Akademie Verlag, Berlin, 1996, pp. 186–192.

[34] M.N. Vrahatis, *Simplex bisection and Sperner simplices*, Bull. Greek Math. Soc., 44 (2000), pp. 171–180.

[35] 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.

[36] M.N. Vrahatis, O. Ragos, T. Skiniotis, F.A. Zafiropoulos, and T.N. Grapsa, *RFSFNS: A portable package for the numerical determination of the number and the calculation of roots of Bessel functions*, Comput. Phys. Comm., 92 (1995), pp. 252–266; erratum, Comput. Phys. Comm., 117 (1999), p. 290.

[37] 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.