Expected Behavior of Bisection Based Methods for Counting and Computing the Roots of a Function

D.J. KAVVADIAS, F.S. MAKRI, M.N. VRAHATIS
Department of Mathematics, University of Patras,
GR-261.10 Patras, GREECE, and
University of Patras Artificial Intelligence Research Center–UPAIRC,
University of Patras, GR-261.10 Patras, GREECE.

URL:

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 number of roots of the order of the hundreds. The method draws its power from the fact that the roots are expected to be many, in order to discover a large proportion of them very efficiently. Its main advantage, apart from its efficiency, is its simplicity which makes it a suitable preprocessing step to more robust and expensive methods in order to reduce the size of the problem. The algorithm is accompanied by a probabilistic analysis of its behavior, which shows that a simple existence criterion like the Bolzano's rule, can be a powerful tool in the root finding process.

Keywords and phrases: Expected behavior, bisection based methods, counting and computing the roots of a function, very large problems.

1 Introduction

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

$$f(x) = 0, \tag{1}$$

where $f : [a, b] \subset \mathbb{R} \to \mathbb{R}$ is continuous in the given interval $[a, b]$.

The method is focused on very large instances of the problem (with roots of the order of the hundreds or more). It is well known that problems of such size emerge in several fields of mathematics and engineering. On the other hand, it is also well known that solving such instances is a formidable task and in some sense the true testing field for any root finding method.

The proposed method takes advantage of the abundance of roots, in an attempt to very cheaply (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 the 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 function evaluations, the problem can be considerably reduced. But this is already more than it 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. Then the unsearched parts of the interval must be searched by a different method. Thus the algorithm must be viewed as a preprocessing step of a more robust and expensive method. Its main advantage 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 would like to point out here that recently, there has been a surge of interest concerning the expected behavior of numerical algorithms [1, 2, 3, 14] and [5, 6, 8, 19]. 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.

The analysis presented here follows the framework of [8].

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 in order to a priori evaluate the size of a problem or as in this case, to establish a stopping criterion for the method.

In the next section we give some background material on the bisection method, as was modified in [15, 16]. In section 3 we briefly discuss the main steps of the algorithm in order to present its theoretical analysis which follows in section 4. Then in section 5 we give a more detailed description of the method which also refers to the theoretical analysis. We end in section 6 with 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 [17]). 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 on the existence or not of a root (see e.g. [9, 10, 11]). 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 [4, 12]. This oracle was used in [8] as part of the first phase of an algorithm for the isolation of all simple roots of a function $f(x)$ in an interval (a, b).

The algorithm of [8] uses in its second phase, bisection in order to compute the roots. Specifically, it uses the following simplified version described in [15]:

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

where $c = \operatorname{sgn} f(a)\,(b - a)$. The sequence (2) converges to a root $r \in (a, b)$ if for some x_i,

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

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

$$\nu = \lceil \log(b - a)\, \varepsilon^{-1} \rceil, \qquad (3)$$

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) we can also use the following one:

$$x_{i+1} = x_i - c\, \operatorname{sgn} f(x_i)/2^{i+1}, \quad i = 0, 1, \ldots, \qquad (4)$$

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

The reason for choosing the bisection method is that it always converges within the given interval (a, b) and it is a globally convergence method. Moreover it has a great advantage since it is optimal, i.e. it possesses asymptotically the best possible rate of convergence [13]. Also, using the relation (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, it requires only the algebraic signs of the function values to be computed, as it is evident from (2) or (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 [18, 20] that the sign stabilizes after a relatively small number of terms of the series and the calculations can be speed up considerably.

3 An informal description

Here is an informal description of the algorithm. The algorithm begins by subdividing the interval into a number of equal subintervals (it will become clear that it is convenient for the number of subintervals to be a power of two and its specific value will be fixed later). The main body of the algorithm consists of three steps:

Step 1. We compute the sign of the function at the endpoints of the subintervals. Depending on the number of roots in each subinterval, its endpoints will have the same signs (in a case of even roots) or opposite signs (in a case of odd roots). We therefore have certainty on the existence of at

least one root in an interval with opposite signs, so we proceed in discovering a root in those intervals using ordinary bisection.

Step 2. Notice that at this point all intervals in our pattern of subdivisions have the same sign at their endpoints. Depending now on a stopping criterion, we decide whether it is worthwhile continuing the algorithm (in terms of the cost of discovering new roots).

Step 3. If we decide that it is, we subdivide the class of subintervals of greatest length into two halves and go back to step 1. Otherwise we output the discovered roots and halt.

4 Work estimations

Our stopping criterion is based on the expected, per root, number of function evaluations that the roots that will be discovered in the next iteration, will require. The probability space on which we make our calculations and the whole framework can be found in [8]. We briefly mention here that we view each root as a random point in the interval (a, b). We also make the assumption that the roots are *uniformly* distributed, i.e. intervals of equal length have equal probability of containing a root. Central to our decision turns out to be an estimation of the total number of roots in the original interval. This is of no surprise: if there is an abundance of roots in the interval and we have only discovered a few, then it is natural to expect that with high probability the next iteration will reveal a lot of roots. The estimation on 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 on 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 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:*

$$p_{\text{odd}} = \frac{1-(1-2\ell)^N}{2}. \tag{5}$$

Proof: See [7].

Throughout this paper we assume without loss of generality that the given interval is normalized, i.e. its length is 1. Consequently the length ℓ of a subinterval gives also the probability of a root to lie within this subinterval. Notice that for a number of roots N of the order of twenty or so, p_{odd} is very close to $1/2$ for small enough ℓ. Therefore, roughly half of the intervals provably have at least one root. This is natural, since when the number of roots is large, the intervals with odd and the intervals with even roots are 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, the number of intervals with zero roots begin to increase, and this is precisely what we take advantage of to decide if it is time to stop. In order however to quantify the above, we use a known confidence interval for the probability p of the binomial distribution. We may view the family of subintervals of interest as random variables which follow, for the property of having or not odd roots, the binomial distribution with probability given by (5).

Proposition 2 *Assume that at a certain point of subdivision we have m subintervals of largest size, out of which we have observed k subintervals with opposite signs at their endpoints. Then with probability at least $1-\alpha$, the probability of having odd roots p_{odd} is between:*

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

where:

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

and

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

Proof: See [7].

In the above formulas $z_{\alpha/2}$ is a constant which can be determined from:

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

where Z is a random variable following the standard normal distribution. For example when $\alpha = 0.05$ then $z_{\alpha/2} = 1.96$.

We use the above confidence interval to estimate p_{odd} and then using (5) to estimate N. This is done by computing two values for N, call them N_{lower} and N_{upper}. The first is computed by solving the equation (with respect to N):

$$p_{\text{upper}} = \frac{1-(1-2\ell)^N}{2}, \tag{6}$$

and the second by solving the equation:

$$p_{\text{lower}} = \frac{1-(1-2\ell)^N}{2}. \tag{7}$$

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 as N to be N_{lower}. Whichever our choice might be, we use it to decide whether it is time to stop. Our criterion is based on the expected work (in function evaluations) that the next root that will be discovered, will require.

Assume that at a certain point our estimation for the total number of roots is N. Moreover, the length of the greatest subintervals is ℓ and their number is m. (Notice that if this happens at the i-th iteration, $\ell = 2^{-i}$).

Now, the work that the algorithm will do at the next iteration consists of (a) the m function evaluations at the midpoints of the m largest subintervals and (b) if k sub-subintervals are discovered with odd number of roots, $k\log(\ell\epsilon^{-1})$ function evaluations for discovering a root in these k subintervals. The total cost therefore of the next iteration is:

$$m + k\log\frac{\ell}{\epsilon}. \tag{8}$$

Now, since the probability of having k sub-subintervals with odd number of roots is:

$$\binom{m}{k} p_{\text{odd}}^k (1-p_{\text{odd}})^{m-k},$$

we get that the expected cost of the next iteration is:

$$\mathrm{E}_{\text{cost}} = \sum_{k=0}^{m}\left[m + k\log\frac{\ell}{\epsilon}\right]\binom{m}{k} p_{\text{odd}}^k (1-p_{\text{odd}})^{m-k}.$$

Since:

$$\sum_{k=0}^{m}\binom{m}{k} p_{\text{odd}}^k (1-p_{\text{odd}})^{m-k} = 1,$$

and

$$\sum_{k=0}^{m} k\binom{m}{k} p_{\text{odd}}^k (1-p_{\text{odd}})^{m-k} = mp_{\text{odd}},$$

we get the following:

$$\mathrm{E}_{\text{cost}} = m + mp_{\text{odd}}\log\frac{\ell}{\epsilon}.$$

Taking into consideration that the expected number of roots of the next iteration is mp_{odd} we get that the expected cost *per root*, $\mathrm{E}^*_{\text{cost}}$, of the next iteration is:

$$\mathrm{E}^*_{\text{cost}} = \frac{1}{p_{\text{odd}}} + \log\frac{\ell}{\epsilon}. \tag{9}$$

The above expected cost constitutes our stopping criterion. The most natural choice is to stop once this cost as given by (9) exceeds some predefined threshold c_{th}. The intention is to stop computing roots using the current method if the cost of discovering new roots becomes very high. But there are also other approaches to this end: for example we may choose to end the algorithm once a predefined "budget" of function evaluations has expired. Current work involves experiments for evaluating c_{th} and the different methods for stopping.

5 The algorithm

In this section we give a detailed description of the algorithm based on the previous discussions.

1. Divide the interval in 32 equal subintervals;
2. Let A be the set of subintervals with opposite signs and B the set of subintervals with the same signs at their endpoints;
3. Find one root in each interval in A using bisection;
4. Estimate the total number of roots using Relations (6) and (7);
5. If both N_{lower} and N_{upper} are finite compute the expected cost of new roots using Relation (9);
6. If the cost is less than the threshold or at least one of the N is infinite then divide the subintervals in B and go to Step **2**.
7. If the condition in Step **6** fails then output the roots and halt.

The initial subdivision in 32 subintervals is forced by the theory supporting the confidence interval in Proposition 2. In order for this interval to hold, m has to be greater than 30.

The subdivision into intervals which are fractions that are 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 ordinary 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 these subdivisions that emerge from bisection, can be used in the future should the algorithm proceeds to next iterations and thus some function evaluations can be saved.

6 Conclusion

We have addressed the problem of computing the roots of a function in a case where the number of roots is very large. This is a formidable problem with many applications in various fields of science. It seems however possible to attack the problem by taking advantage precisely 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 severely 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. Current research includes experimental evaluation of a number of parameters in order to better tune the algorithm.

References

[1] S. Graf, R.D. Mauldin, S.C. Williams, "Random homeomorphisms", Adv. Math., vol.60, 1986, pp.239–359.

[2] S. Graf, E. Novak, "The average error of quadrature formulas for functions of bounded variation", Rocky Mountain J. Math.

[3] S. Graf, E. Novak, A. Papageorgiou, "Bisection is not optimal on the average", Numer. Math., vol.55, 1989, pp.481–491.

[4] B.J. Hoenders, C.H. Slump, "On the calculation of the exact number of zeros of a set of equations", Computing vol.30, 1983, pp.137–147.

[5] D.J. Kavvadias, F.S. Makri, M.N. Vrahatis, "Locating and computing arbitrarily distributed zeros and extrema", 4th Hellenic European Conference on Computer Mathematics and its Applications, E.A. Lipitakis Ed., 1998, in press.

[6] D.J. Kavvadias, F.S. Makri, M.N. Vrahatis, "Locating and computing arbitrarily distributed zeros", SIAM J. Sci. Comput., in press.

[7] D.J. Kavvadias, F.S. Makri, M.N. Vrahatis, "Expected behavior of bisection based methods for counting and computing the roots of a function", Division of Computational Mathematics and Informatics, Department of Mathematics, University of Patras, Technical Report TR/1999-1, 1999.

[8] D.J. Kavvadias, M.N. Vrahatis, "Locating and computing all the simple roots and extrema of a function", SIAM J. Sci. Comput., vol.17, no.5, 1996, pp.1232–1248.

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

[10] R.B. Kearfott and M. Novoa III, "INTBIS: A portable interval Newton/bisection package", ACM Trans. Math. Software, vol.16, 1990, pp.152–157.

[11] A. Neumaier, "Interval Methods for systems of equations", Cambridge University Press, Cambridge, 1990.

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

[13] K. Sikorski, "Bisection is optimal, Numer. Math. vol.40, 1982, pp.111–117.

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

[15] M.N. Vrahatis, "Solving systems of nonlinear equations using the nonzero value of the topological degree", ACM Trans. Math. Softw., vol.14, 1988, pp.312–329.

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

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

[18] M.N. Vrahatis, T.N. Grapsa, O. Ragos, F.A. Zafiropoulos, "On the localization and computation of zeros of Bessel functions", Z. angew. Math. Mech., vol.77, 1997, pp.467–475.

[19] M.N. Vrahatis, D.J. Kavvadias, "Expected behavior of bisection based methods for counting and computing all the roots of a function", Proceedings of the Sixth International Colloquium on Differential Equations, D. Bainov, Ed., VSP International Science Publishers, Zeist, The Netherlands, 1996, pp.353–360.

[20] M.N. Vrahatis, O. Ragos, F.A. Zafiropoulos and T.N. Grapsa, "Locating and Computing Zeros of Airy Functions", Z. angew. Math. Mech., vol.76, 1996, pp.419–422.