Applied Mathematics and Computation 268 (2015) 628–635

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Global root bracketing method with adaptive mesh refinement

M.A. Razbani*

Department of Chemical Engineering, Technical and Vocational University, Jajarm Branch, Jajarm, Iran

ARTICLE INFO

Keywords:
Root finding
Bracketing methods
Bisection
Adaptive mesh refinement
Odd-multiple roots
Even-multiple root

ABSTRACT

An efficient method for finding all real roots of a univariate function in a given bounded domain is formulated. The proposed method uses adaptive mesh refinement to locate bracketing intervals based on bisection criterion for root finding. Each bracketing interval encloses one root. An adaptive form of error is introduced to enclose roots in a desired tolerance based on how close the roots are. Detecting roots with even multiplicity, which is regarded as beyond the realm of bracketing methods, becomes possible with the method proposed in this paper. Also, strategies for finding odd-multiple roots with the least number of function evaluations are proposed. Adaptive mesh refinement lead to considerable reduction in function evaluations in comparison to previous global root bracketing methods. The reliability of the proposed method is illustrated by several examples.

1. Introduction

A one dimensional root finding problem finds x such that $f(x) = 0$, for a given function $f : R \to R$. Such an x is called a root of the equation $f(x) = 0$ or a zero of the function f. Numerical method algorithms which deal with solving this problem can be divided into two basic groups, bracketing methods and open methods. Bracketing methods start with a bounded interval which is guaranteed to bracket a root. The size of the initial bracket is reduced step by step until it encloses the root within a desired tolerance. Open methods begin with an initial guess of the root and then improve the guess iteratively. Bracketing methods provide an absolute error estimate on the root's location and always work but converge slowly. In contrast, open methods do not always converge. There is a trade-off between absolute error bound and speed in open methods. Bracketing methods use function evaluations but some open methods need both the function and its derivatives.

There are at least three major kinds of bracketing methods. The first one is the bisection method. The second one is false position which is a method of finding roots based on linear interpolation. The third one is the Brent-Dekker method which combines an interpolation strategy with the bisection algorithm. The bisection method or interval halving is the simplest bracketing method for root finding of a continuous non-linear function, namely $f(x)$. This method has a linear convergence rate [1]. The first step in the bisection method is to provide a search bound. The search bound represented by $[x_a\ x_b]$ is the limit of an interval where the sign of $f(x_a)$ and $f(x_b)$ are different. Based on the Intermediate Value Theorem, when $f(x_a)$ and $f(x_b)$ have opposites signs, then there is at least one real root between x_a and x_b. The iteration begins with halving the search space. The midpoint of the interval $x_m = (x_a + x_b)/2$ and $f(x_m)$ are evaluated. If $f(x_a) \times f(x_m) < 0$, then x_b is replaced by x_m otherwise x_a is replaced by x_m. The search space is halved at each step. The process is repeated and the root estimate refined by dividing the subintervals into finer increments. Iteration terminates if at any point $f(x)$ equals 0 or the distance between x_a and x_b becomes lower than a requested precision. This well-known algorithm can find only one root in the search bound. If there is more than one root, it is

* Tel.: +98 9153854817.
E-mail address: amin.razbani@gmail.com

http://dx.doi.org/10.1016/j.amc.2015.06.121
0096-3003/

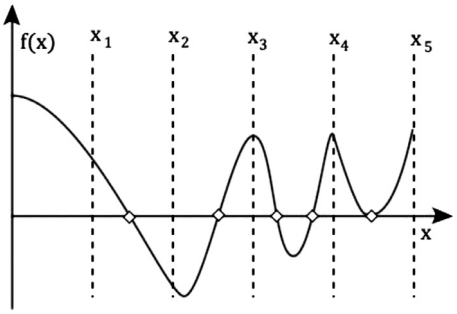

Fig. 1. Cases where roots could be missed in GRBM.

unclear which root is found. It's very favorable to extend bracketing methods to somehow find all roots of an objective function in a given interval. Such a method is named a "global root bracketing method" (GRBM). In this paper a globalization of the bisection method for finding all roots of an objective function is proposed.

The rest of the paper is organized as follows. In Section 2, the GRBM as previously proposed will be reviewed and its deficiencies will be shown. In Section 3, a new approach which adds an adaptive mesh refinement (AMR) feature to the GRBM is introduced. It will be shown that implementing AMR can greatly increase the speed of GRBM by adding new features, and suppressing the deficiencies. Finally, a conclusion is given in Section 4. In this paper, root finding is discussed for one-dimensional problems but the idea can be applied to higher dimensions and that will be discussed in further publications.

2. Global root bracketing method

Using bracketing methods for finding all the roots of an objective function has been previously reported [2–8]. Some papers referred to the univariate case [6,8] and some others to the multi-dimensional case [2–5,7]. For a detailed review of root bracketing methods refer to [1]. Such methods are sometimes called bisection-exclusion methods. In the GRBM, an incremental search approach is used to locate subintervals where the function changes sign. These subintervals are called bracketing intervals. Other subintervals, which fail to pass this bisection criterion, are excluded. A bracketing method is used to locate the root in every one of the bracketing intervals. The generic form of GRBM can be presented in the following algorithm [1]:

1. Input: $f(x)$, search bound limits, halving threshold (HT), tolerance for stopping criterion
2. Create an initial mesh
3. Evaluate $f(x)$ at the nodes of the initial mesh
4. Select bracketing intervals
5. Apply a root bracketing method to each bracketing interval
6. Output roots founded in step 5

In some works, the search interval is divided into sub-intervals all at once [5]. In some other works, dividing goes on step by step [2]. If an interval is not detected as a bracketing interval and its size is bigger than the HT, it should be halved. That's because failure in fulfilling the bracketing criterion doesn't guarantee that a subinterval does not contain any root. So, it should be further reduced in size. The HT is chosen to limit the halving process. The problem which arises is that a small HT is computationally costly. On the other hand, if a large HT is chosen, some roots may be omitted. The problem is compounded by the possible existence of even-multiplicity roots, such as $f(x) = (x - x_0)^2$. A representation of GRBM is depicted in Fig. 1.

Dashed lines in Fig. 1 show increment length of the mesh. The distance between any of the two adjacent dashed lines is the HT. One can see that the HT is not small enough to detect bracketing intervals which enclose either of the two roots which lie in $[x_3\ x_4]$. The last root on the right is a root with even-multiplicity and would be missed regardless of HT because the curve never crosses the x axis at an even-multiple root. Roots which lie in $[x_1\ x_2]$ and $[x_2\ x_3]$ can be detected correctly. It's clear from Fig. 1 that GRBM with static mesh refinement leads to constant grid spacing.

Here, the first deficiency of the generic form of GBM is discussed. There are regions in the search space which need to be investigated with smaller HT ($[x_4\ x_5]$ in Fig. 1). On the other hand, there are regions ($[x_1\ x_2]$ in Fig. 1) where a large HT can take care of root finding. Uniform mesh refinement which is depicted in Fig. 1 fails to find bracketing intervals efficiently. In order to separate close roots with static mesh refinement, HT should not be bigger than the minimum distance between roots. In Section 2, an adaptive mesh refinement will be presented which can solve this problem.

The other problem is related to the definition of error or termination criterion in root bracketing methods. The acceptable form of error for root bracketing methods including the bisection method is the same as in other iterative methods and can be expressed as follows [9]:

$$\varepsilon_a \equiv \left| \frac{x_m^{new} - x_m^{old}}{x_m^{new}} \right| \tag{1}$$