Method Article

A new method for rooting nonlinear equations based on the Bisection method

Reza Etesami\textsuperscript{a}, Mohsen Madadi\textsuperscript{a,}\textsuperscript{*}, Mashaallah Mashinchi\textsuperscript{a}, Reza Ashraf Ganjoei\textsuperscript{b}

\textsuperscript{a}Department of Statistics, Faculty of Mathematics & Computer, Shahid Bahonar University of Kerman, Kerman, Iran
\textsuperscript{b}Department of Economics, Shahid Bahonar University of Kerman, Kerman, Iran

\textbf{A B S T R A C T}

Finding the roots of nonlinear equations has many applications in various sciences, especially engineering, and various methods have been proposed for this purpose. However, almost all these methods have some shortcoming. This paper presents a new method, where we consider the desired function to find the root(s) of the absolute value, so the root(s) (if any) is the absolute minimum. Using Monte Carlo method, we divide the desired distance into smaller parts. In each section where the slope of the function changes, we use the Bisection method to find the root. It largely covers the limitations of previous methods. The most important advantage of this method over the Bisection method is that it finds all the roots of the equation.

- Solve the problem of the bisection method in roots tangent to the x-axis.
- Separation of Root(s) that crossed and Root(s) that are tangent to the x-axis.

© 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

\textbf{A R T I C L E   I N F O}
Method name: Improved Bisection method based on slope (IBMS)
Keywords: Finding multiple roots, Nonlinear algebraic equation, Bisection method, R Software
Article history: Received 8 February 2021; Accepted 25 August 2021; Available online 26 August 2021

\* Corresponding author.
E-mail addresses: madadi@uk.ac.ir (M. Madadi), mashinchi@uk.ac.ir (M. Mashinchi), ashrafganjoei@aem.uk.ac.ir (R.A. Ganjoei).

https://doi.org/10.1016/j.mex.2021.101502
2215-0161/© 2021 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Specifications table

| Subject Area:        | Engineering |
|----------------------|-------------|
| More specific subject area: | Rooting Nonlinear Equations |
| Method name:         | Improved Bisection method based on slope (IBMS) |
| Name and reference of original method: | 1.He, J. H. (1998). Newton-like iteration method for solving algebraic equations, Commun. Non-linear Sci. Numer. Simulation, 3(2). |
|                      | 2.Rizzo, M. L. (2008). Statistical computing with R. |
|                      | 3.Yamamoto, T. (2000). Historical developments in convergence analysis for Newton’s and Newton-like methods. Journal of Computational and Applied Mathematics, 124(1–2), 1–23. |
|                      | 4.Jones, O., Maillardet, R., & Robinson, A. (2014). Introduction to scientific programming and simulation using R. CRC Press. |
| Resource availability: | https://www.r-project.org/ |

Introduction

Why are we interested in nonlinear problems? Perhaps the most important reason is that all of those problems are nonlinear from the beginning when a phenomenon is under study. However, the linearization commonly employed are an approximating device arising from our confession that we are incapable of facing the challenge presented by the nonlinear problems as such linearization [2].

Many physical problems in basic sciences and engineering are modeled in the form of nonlinear equations. In this regard, solving them is a challenge in various sciences, so it is easier to use numerical methods. The study of solving nonlinear equations is an important part of applied mathematics. Because the most of real-world phenomena can be modeled according to a nonlinear equation or systems of nonlinear equations. Since finding the root(s) in a nonlinear equation is very difficult and sometimes impossible, so we often use repetitive numerical methods to get the root(s). Nowadays the advances in computers and the growth of applied software’s can solve many complex and difficult problems as well as nonlinear equations [1].

Therefore, methods such as artificial neural network [3], Monte Carlo [2], fuzzy logic [6] are used to find the root(s) of nonlinear equations and are able to cover some of the shortcoming of previous methods such as Newton Raphson [5], Bisection Method [4], False Position [5], Secant Method [7].

Some of the shortcoming of the proposed methods are: Dependence on initial value (S1), It may not converge(S2), Relies on sign changes (If a function \( f(x) \) is such that it just touches the x-axis for example say \( f(x) = x^2 \) then it will not be able to find lower guess (a) such that \( f(a) \ast f(b) < 0 \) (S3), Costly calculations (S4), Cannot detect Multiple Roots (S5).

In this paper, the Monte Carlo and Bisection method is used to find the root(s) of nonlinear equations which largely covers the limitations of known existing methods. The organization of this research is as follows: In section 2, a summary of previous methods is presented. In Section 3, we describe a Monte Carlo Based Approach for finding Multiple Roots of Nonlinear Equations. Section 4 presents the Result and Discussion. Section 5 presents the conclusions. Table 1.

| Table 1 |
|---------|
| Comparison of the disadvantages of rooting methods. |

| Shortcoming                  | S1 | S2 | S3 | S4 | S5 |
|------------------------------|----|----|----|----|----|
| Bisection Method             | ✗  | ✗  | ✓  | ✗  | ✓  |
| Newton Raphson Method        | ✓  | ✓  | ✗  | ✓  | ✓  |
| False Position               | ✗  | ✗  | ✓  | ✗  | ✓  |
| Secant Method                | ✗  | ✓  | ✗  | ✓  | ✓  |
| New Method (IBMS)            | ✗  | ✗  | ✗  | ✓  | ✗  |
Summary of other methods

Bisection method

One of the simplest algorithms for finding algebraic equations is the bisection method. This method can be used for any continuous function \( f \) on in the domain \([a, b]\). According to Intermediate Value Theorem, if \( f(a).f(b)\) have opposite signs, there is at least one root in this range. To finding the root, the range \([a, b]\) is divided into two parts \(c = (a + b)/2\). If \( f(a).f(c)\) is negative, the domain will be restricted by \([a, c]\) and otherwise it is replaced by \([a, c]\). In each iteration, the length of the distance, which contain root, is halved [4].

Some disadvantages: (S3) and (S5) [8].

Newton Raphson method

One of the numerical methods in which derivation is used and applied in various fields is Newton's method. It is a numerical and approximate method for solving equations that can converge faster than the bisection method [5].

Some disadvantages: (S1), (S2), (S4) and (S5).

False position method

The False Position Method, which also known as the Regula False Method is similar to the bisection method. The only difference is that it converges faster than the bisection method. In this method, instead of halving the distance \([a, b]\), the distance \([a, b]\) is reduced by using a chord that connects the points \(f(a)\) and \(f(b)\). The point where the chord meets the x-axis is called \(c\). If \(f(a).f(c)\) is negative, the domain will be restricted by \([a, c]\) and otherwise it is replaced by \([c, b]\) [9].

Some disadvantages: (S3) and (S5).

Secant method

This method differs from Newton’s method in that it does not require a derivative of a function. It is also not necessary that our two starting points be on either side of the root of the function. This method does not guarantee convergence. But if it is convergent, it will quickly approach the root [5].

Some disadvantages: (S2) and (S5).

Algorithm for finding multiple roots of nonlinear equations

Using Monte Carlo method, we divide the desired distance into smaller parts. In each section that specifies the change function, we use the Bisection method to find the root. Extreme points close to the y-axis are used to find the tangent root to the x-axis. Algorithm codes in R software are provided in the appendix.

Step 1:
Generating random sample numbers \(x\) from a uniform distribution in the interval \([a, b]\).

Step 2:
\[ \exists\text{ s.t.} \ f(x_i) f(x_{i+1}) < 0 \] (Like the green dots in Fig. 1).
Finding \(x_i\).

Step 3:
We use the Bisection method to find the root at distance \([x_i, x_{i+1}]\).

Step 4:
\[ (f(x_i + \varepsilon) - f(x_i))(f(x_{i+1} - \varepsilon) - f(x_{i+1})) < 0 \]
To find the next root, we repeat the first to third steps with \( x_{i+1} \). Until all the roots are found in the distance \([a, b]\).

**Step 5:**
For root(s) that are tangent to the x-axis:
We consider the absolute value of the function, for root(s):

\[
\exists x_i \text{ s.t. : } (f(x_i + \varepsilon) - f(x_i))(f(x_{i+1} + \varepsilon) - f(x_{i+1})) < 0 \text{ (Like the blue dots in Fig. 1).}
\]
Finding \( x_i \).

**Step 6:**
We use the Bisection method to find the root at distance\([x_i, x_{i+1}]\).

**Step 7:**
To find the next root, Repeat steps five through seven with \( x_{i+1} \). Until all the roots are found in the distance\([a, b]\).

**Step 8:**
Finally, we examine the beginning and end points of the interval\([a, b]\).

**Result and discussion**

The advantage of the proposed method is finding Multiple Roots of Nonlinear Equations without have dependency on the initial value and Relies on sign changes. In the previous numerical methods, solving equations of degree \( n \), the answers must be calculated one by one and therefore the answers before and after are interdependent, but the proposed method is able to obtain all the roots of the equation at the same time. So this dependence disappears the advantages of this method can almost cover the disadvantages of other methods. Also in this method, after finding the root(s) of the equation, the root(s) tangent to the x-axis and the root(s) intersecting the x-axis are separated.

**Conclusion**

In this paper, we used the integration of Monte Carlo method and Bisection method to find the root(s) of nonlinear equations. The approximate position of the root(s) is done by examining a random sample of numbers generated in places where the change function signals and where there are extreme points near the y-axis and convergence to the root using the Bisection method. Hence, the dependence on the initial value disappears and all the roots(s) of the equation can be found at one time. Given the advantages of the proposed method over previous methods, it can be used as a new method in solving algebraic equations.
Declaration of Competing Interest

The authors declare that they have no known competing for financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

R program to find the root(s) with the proposed method:

```r
RootMC <- function(f,a,b) {
  n1 = c()
  n2 = c()
  LE = b-a
  ###Step 1:
  x = runif(2000,a,b)
  eps <- .Machine$double.eps^0.5
  t0 = length(x)
  E = t0 - 1
  x = sort(x)
  for (i in 1:E) {
    ###Step 4:
    A = f(x[i])
    B = f(x[i+1])
    if ((A*B)<0) {
      b0 <- x[i]
      b1 <- x[i+1]
      ###Step 3:
      it <- 0
      r <- seq(b0, b1, length=3)
      y <- c(f(r[1]), f(r[2]), f(r[3]))
      while (it < 1000 && abs(y[2]) > eps) {
        it <- it + 1
        if (y[1]*y[2] < 0) {
          r[3] <- r[2]
          y[3] <- y[2]
        } else {
          r[1] <- r[2]
          y[1] <- y[2]
        }
        r[2] <- (r[1] + r[3]) / 2
        y[2] <- f(r[2])
      }
      if (abs(f(r[2])) < (10^-3)) {
        n1 = c(n1,r[2])
      }
    }
    T4 = length(n1)
    if(T4>0){
      n1 = round(n1, digits = 10)
      n1 = matrix(n1, T4, 1, dimnames =
                  list(c(paste(c("Root"), 1:(T4), sep="")),
paste("Root(s) that crossed the x-axis:"))))
      print(n1)
    } else {
      print("There is no root that has crossed the x-axis")
    }
  }
  #root(s) that are tangent to the x-axis:
```
for (i in 1:E) {###Step y:
  A=f(x[i])
  B=f(x[i]+1)
  AA=f(x[i])+eps*10^5-A
  BB=f(x[i]+1)+eps*10^5-B
  if (AA*BB<0){###Step 5:
    b0 <- x[i]
    b1 <- x[i+1]
    ###Step 6:
    it <- 0
    r <- seq(b0, b1, length=3)
    y <- c((f(r[1])+eps*10^3)-f(r[1]),(f(r[2])+eps*10^3)-f(r[2])),
          (f(r[3])+eps*10^3)-f(r[3]))
    while(it < 1000 & abs(y[2]) > eps) {
      it <- it + 1
      if (y[1]*y[2] < 0) {
        r[3] <- r[2]
        y[3] <- y[2]
      } else {
        r[1] <- r[2]
        y[1] <- y[2]
      }
      r[2] <- (r[1] + r[3]) / 2
      y[2] <- (f(r[2])+eps*10^3)-f(r[2])
    }
    if(abs(f(r[1])) < (10^3)){
      n2=c(n2,r[1])
    }
  }
  ###Step 8:
  if(abs(f(a)) < 10^-10){n2=c(n2,a)
  }
  if(abs(f(b)) < 10^-10){n2=c(n2,b)
  }
  T4=length(n2)
  if(T4>0){
    n2=round(n2,digits = 10)
    n2=matrix(n2,T4,1,dimnames =
               list(c(paste(c("Root"),1:(T4),sep="")),paste("Root(s)
              that are tangent to the x-axis:"))
    print(n2)
  }else{
    print("There is no root that is tangent to the x-axis")
  }
}
#Ex:
RootMC(function(x) {4*x^(10)-5*x^9
           --10*x^2},-10,10)
RootMC(function(x) {x*cos(x)^2},-20,20)
RootMC(function(x) {exp(x-2)-x^(10)+3*x^(9)},-20,20)
#Ex: $\alpha^2 + x^2 + \frac{2\alpha x}{n-1} = n - 2$, $n = 20$, $\alpha = 0.5$ [4].
  a=0.5
  n=20
RootMC(function(x) {(a^2+x^2+(2*a*x)/(n-1))-(n-2)},0,100)
RootMC(function(x) {-1+x^2},-2,2)
References

[1] Vahid Mirzaei Mahmoud Abadi, Shila Banari Bahnamiri, The approximate calculation of the roots of algebraic equation through Monte Carlo Method, Int. J. Math. Comput. Sci. 2.3 (2016) 64–68.
[2] William F. Ames (Ed.), Nonlinear Ordinary Differential Equations in Transport Processes, Academic Press, 1968.
[3] Freitas, Diogo, Luiz Guerreiro Lopes, and Fernando Morgado-Dias. “A neural network based approach for approximating real roots of polynomials.”
[4] Maria L. Rizzo, Statistical Computing With R, CRC Press, 2019.
[5] Owen Jones, Robert Maillardet, Andrew Robinson, Introduction to Scientific Programming and Simulation Using R, CRC Press, 2014.
[6] M.Hadi Mashinchi, M. Reza Mashinchi, S.M.H. Shamsuddin, A genetic algorithm approach for solving fuzzy linear and quadratic equations, World Acad. Sci. Eng. Technol. 28 (2007) 242–246.
[7] Avram Sidi, Generalization of the secant method for nonlinear equations, Appl. Math. E-Notes 8 (2008) 115–123.
[8] Ali. Demir, Trisection method by k-Lucas numbers, Appl. Math. Comput. 198 (1) (2008) 339–345.
[9] Soumen Shaw, Basudeb Mukhopadhyay, An improved Regula falsi method for finding simple roots of nonlinear equations, Appl. Math. Comput. 254 (2015) 370–374.