Universidad Industrial de Santander
Professor: Juan Carlos Basto Pineda
2018- II

Workshop No. 1 - Root-finding Methods
Mathematical modelling
Date: August 29, 2018

Name: Andrés Felipe Jerez Ariza **Scholar ID:** 2188136

1 Introduction

In mathematics and computing, a root-finding algorithm is an algorithm for finding roots of continuous functions. A root of a function f, from the real numbers to real numbers or from the complex numbers to the complex numbers, is a number x such that $f(x) = 0$. As, generally, the roots of a function cannot be computed exactly, nor expressed in closed form, root-finding algorithms provide approximations to roots, expressed either as floating point numbers or as small isolating intervals, or disks for complex roots (an interval or disk output being equivalent to an approximate output together with an error bound).

In order to solve the root-finding problem was developed many algorithms based on bracketing methods, iterative methods and combinations of methods.

- **Bracketing methods:** determine successively smaller intervals (brackets) that contain a root, e.g., bisection method.
- **Iterative methods:** an iterative root-finding method generally use a specific type of iteration, consisting of defining an auxiliary function, which is applied to the last computed approximations of a root for getting a new approximation, e.g., Newton-Rhapson and secant methods.
- **Combinations of methods:** Brent's method is a combination of the bisection method, the secant method and inverse quadratic interpolation. At every iteration, Brent's method decides which method out of these three is likely to do best, and proceeds by doing a step according to that method.

In this work, some root-finding algorithms were implemented such as the bisection method, the secant method, Newton method and Brent's method. Several simulations were carried out in order to evaluate each algorithm according to convergence order, number of iteration, absolute error, and tolerance using four different functions, which are given by

$$f_1(x) = x^2 - 2, \tag{1}$$

$$f_2(x) = x^3 - 10x^2 + 5, \tag{2}$$

$$f_3(x) = e^x \sin(x) + 4, \tag{3}$$

$$f_4(x) = \cosh(x)\cos(x) - 1, \tag{4}$$

In Fig. 1 the sketches of these functions including its roots is presented.

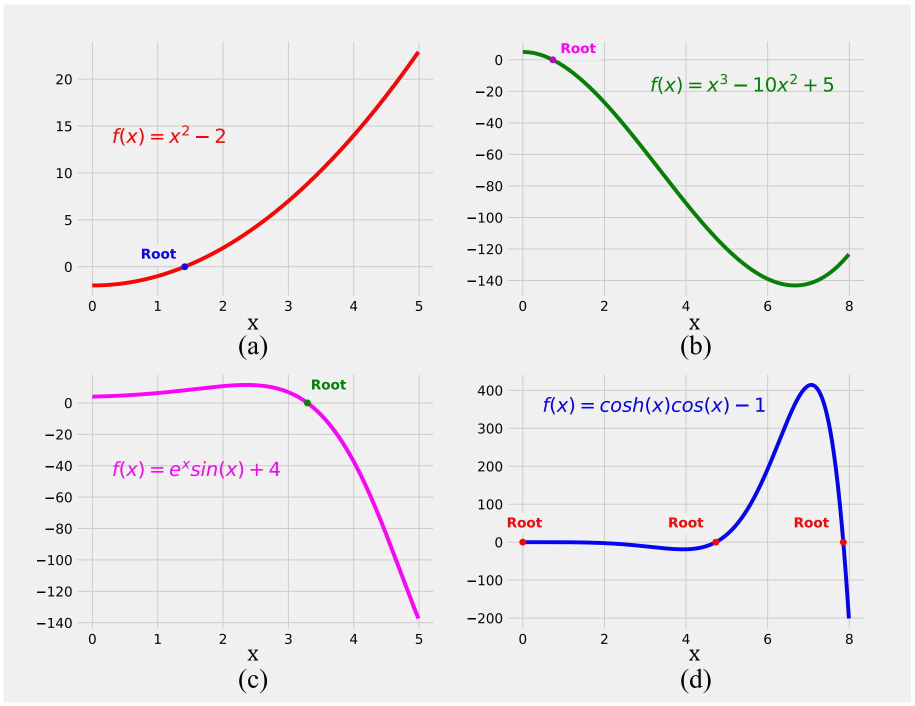

Figure 1: Implemented functions in order to evaluate different root-finding algorithms.

2 Results

In order to evaluate the performance of the bisection method, secant method, Newton method and Brent's method, several simulations were tested. Therefore, absolute error, convergence order, and tolerance were estimated to compare the performance of the implemented algorithms. These metrics are recalled as follow

- Absolute error
$$e_k = |\tilde{x}_k - x|\,, \tag{5}$$
where $\tilde{x}_k \in \mathbb{R}$ represents the approximation root in the k-th iteration and $x \in \mathbb{R}$ is the real root.

- Convergence order
$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^r} = C, \tag{6}$$
where $r \in \mathbb{R}$ corresponds to convergence order and $C \in \mathbb{R}$ is a constant.

- Tolerance:
$$\delta_k = |\tilde{x}_k - \tilde{x}_{k-1}|\,, \tag{7}$$

Notice that the convergence order using Eq. 6 cannot be directly evaluated. Then, Eq. 6 is reorganized using the logarithmic function as

$$e_{k+1} = Ce_k^r \rightarrow \log(e_{k+1}) = \log(C) + r\log(e_k), \tag{8}$$

In table 1 the number of iterations at each root-finding method is presented using a tolerance $\delta = 10^{-4}$. Bisection method requires up to 15 iterations, while Brent's method just requires up to 7 iterations to find a root under this tolerance. In table 2 the convergence order at each root-finding method is shown using a tolerance $\delta = 10^{-4}$. Bisection method requires up to 15 iterations, while Brent's method just requires up to 7. The mean and standard deviation (STD) of the convergence order according to each function and method was calculated. Taken the low difference between experimental and theoretical convergence order into account, the reliability of the simulation results of these methods is can be guaranteed.

	Iterations			
Method	f_1	f_2	f_3	f_4
Bisection	15	15	12	14
Secant	7	7	4	7
Newton	8	6	3	5
Brent	6	7	4	7

Table 1: Amount of iteration at each root-finding method using $\delta = 10^{-4}$.

	Convergence order							
Method	f_1	f_2	f_3	f_4	**Mean**	**STD**	**Real**	**Error**
Bisection	0,7413	0,6860	0,6088	0,8036	0,7099	0,0828	0,5000	0,2099
Secant	1,5987	1,4727	1,6182	1,6249	1,5786	0,0715	1,6180	0,0394
Newton	1,9304	1,9128	1,9978	1,9881	1,9573	0,0420	2,0000	0,0427
Brent	1,8647	1,8446	1,7943	1,8258	1,8324	0,0299	1,8393	0,0069

Table 2: Convergence order at each root-finding method using $\delta = 10^{-4}$.

In Fig 2 the comparison between tolerance δ_k and iteration using a maximum tolerance $\delta = 10^{-4}$ is illustrated. In Fig 3 the comparison between absolute error e_k and iteration using a maximum tolerance $\delta = 10^{-4}$ is presented. On the other hand, in Fig 4 the relationship between e_{k+1} and e_k using Eq. 8 to estimate the convergence order r is shown. Then, the convergence order is given by the slope of the linear fitting according to each root-finding algorithm. Furthermore, in Fig 5 the time measured in seconds of the performance of the tested root-finding methods according to tolerance $\delta = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ is illustrated. Notice that the time increases when we have a low tolerance. However, the Newton method by using Eq. 1 produces a different behavior. In general, the Newton method, secant method, and Brent's method produce an almost constant time according to tolerance.

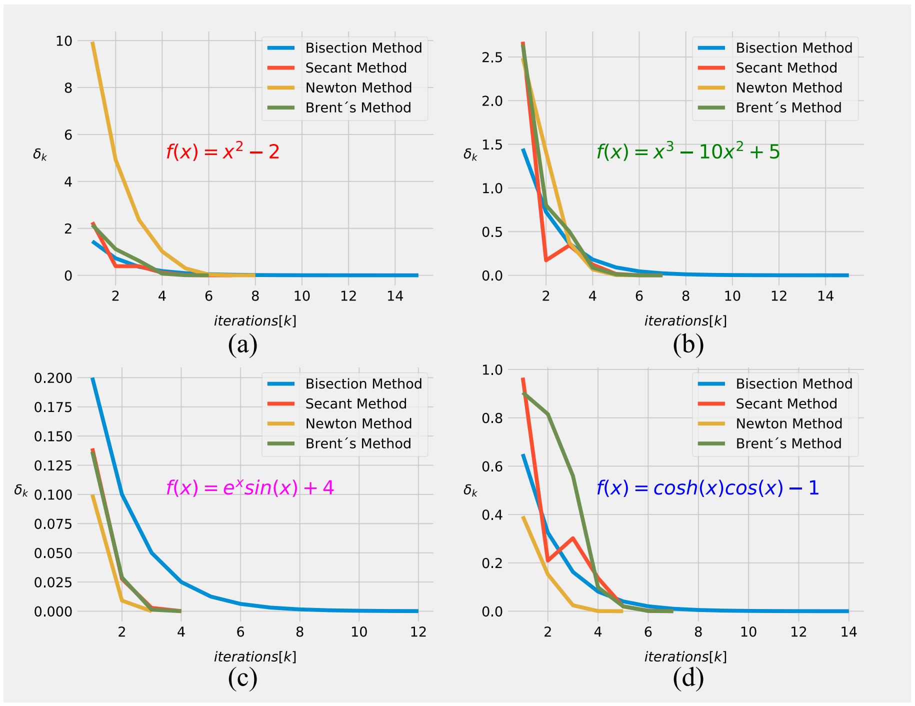

Figure 2: Comparison between δ_k and iteration using different root-finding algorithms and functions.

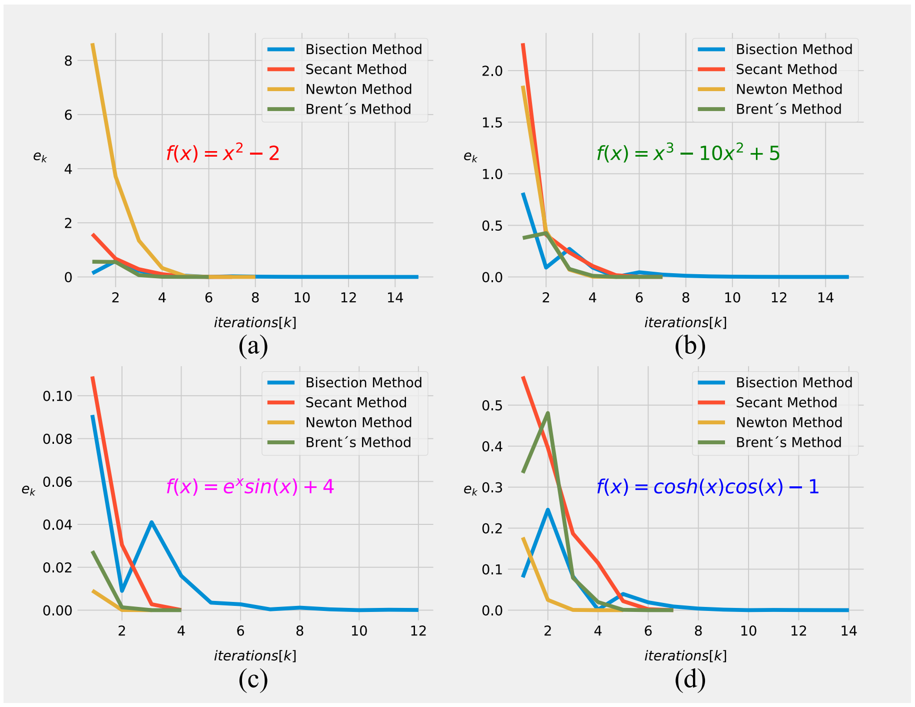

Figure 3: Comparison between e_k and iteration using different root-finding algorithms and functions.

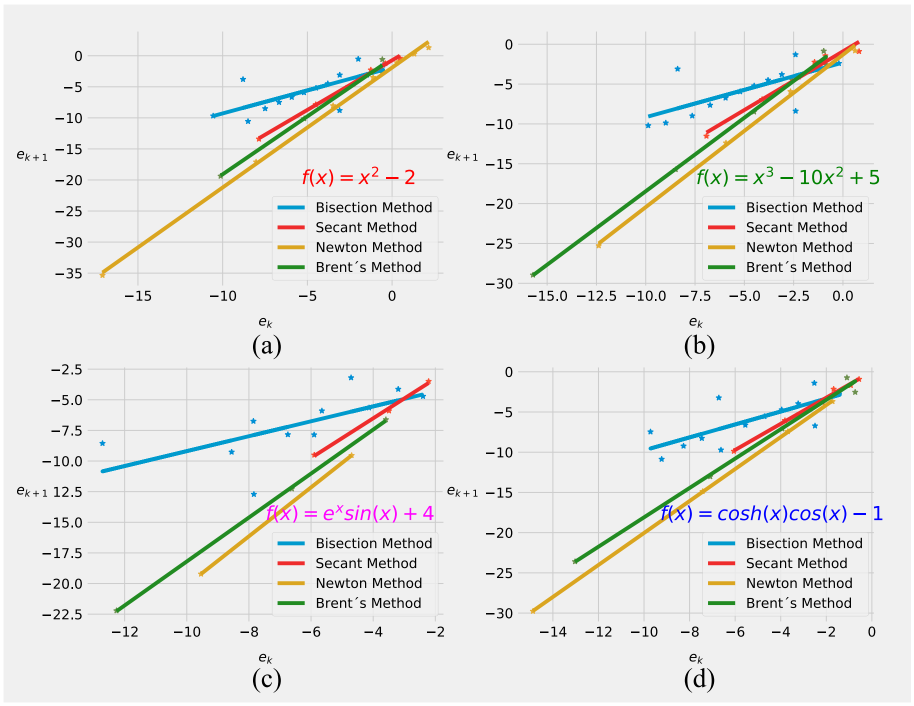

Figure 4: Comparison between the logarithms of e_{k+1} and e_k using different root-finding algorithms and functions.

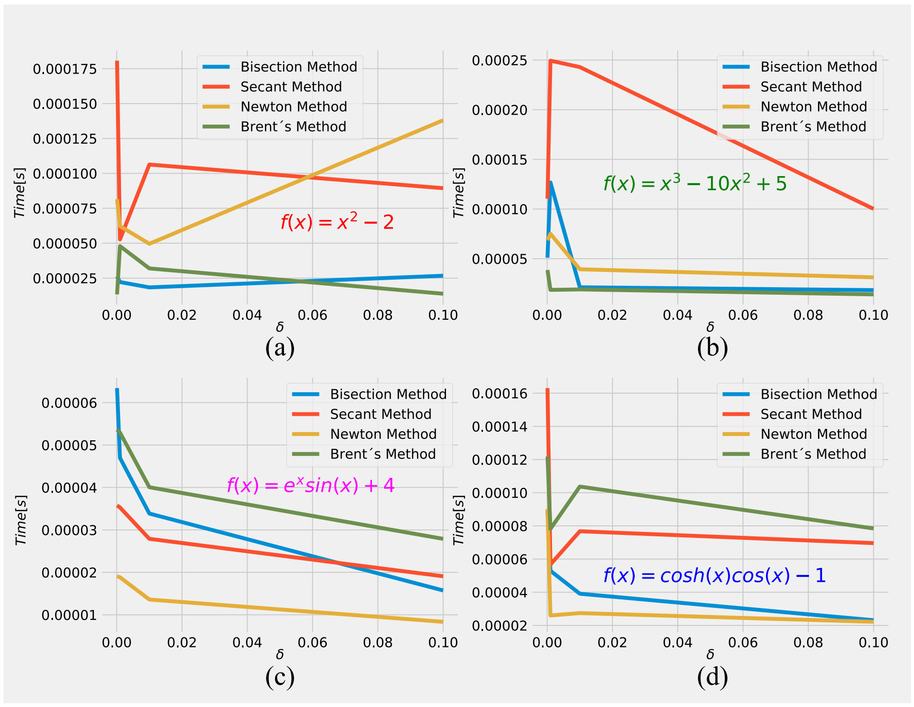

Figure 5: Comparison between perform time and δ using different root-finding algorithms and functions.