Journal of Mathematics Research; Vol. 12, No. 6; December 2020
ISSN 1916-9795 E-ISSN 1916-9809
Published by Canadian Center of Science and Education

An Improved Secant Algorithm of Variable Order to Solve Nonlinear Equations Based on the Disassociation of Numerical Approximations and Iterative Progression

Christian Vanhille

Correspondence: NANLA research group, Universidad Rey Juan Carlos, Tulipán s/n, Móstoles, Madrid 28933, Spain. E-mail: christian.vanhille@urjc.es

Received: September 11, 2020 Accepted: October 15, 2020 Online Published: October 29, 2020

doi:10.5539/jmr.v12n6p50 URL: https://doi.org/10.5539/jmr.v12n6p50

Abstract

We propose an iterative method to evaluate the roots of nonlinear equations. This Secant-based technique approximates the derivatives of the function numerically through a constant discretization step h disassociated from the iterative progression. The algorithm is developed, implemented, and tested. Its order of convergence is found to be h-dependent. The results obtained corroborate the theoretical deductions and evidence its excellent behavior. For infinitesimal h-values, the algorithm accelerates the convergence of the Secant method to order 2 (the one of the Newton-Raphson method) with no need for analytic expression of derivatives (the advantage of the Secant method).

Keywords: nonlinear functions, improved secant method, disassociate numerical iterative algorithm

1. Introduction

Finding the zeros of nonlinear functions has been of interest for a long time (Mathews & Fink, 1999; Dahlquist & Björck, 2008). Regarding graphical and analytic techniques, iterative numerical methods enhance the precision of the results and make it easier for the user to solve practical situations, especially when the nonlinear equations include high-degree terms or transcendental functions.

The Newton-Raphson method was developed in 1669 by Newton, and independently by Raphson in 1690 (Ypma, 1995). Based on the approximation of a root in the iterative sequence from the previous one following the tangent at the curve up to its intersection with the abscissa axis, this algorithm is probably the most used in the world to find the zero of a nonlinear function, from undergraduate and postgraduate students to researchers in sciences and technologies. The potential issue of this technique is that it can be difficult to obtain the analytic expression of the derivative of some functions, which is mandatory to approximate the root at each iteration of the sequence.

Although it was developed in 1665 by Newton, the Secant method is known to overcome this drawback (Ypma, 1995). This adaptation of the Newton-Raphson method changes the analytic evaluation of the derivative of the function for a backward finite-difference formula based on the two previous points obtained in the root sequence. This makes it much easier for the user. However, this substitution leads to a loss in the order of convergence. The quadratic convergence of the Newton-Raphson algorithm drops down to 1.618 (the golden ratio) when substituted by the Secant algorithm.

Since then, many other formula for solving nonlinear equations have been developed, including the ones deduced in 1694 by Halley, all of them with their own positive and negative aspects. Beside Newton-based techniques, Adomian decomposition method, homotopy perturbation method were used to develop numerical schemes for finding the zeros of nonlinear functions (Gander, 1985; Abbasbandy, 2006; Pakdemirli & Boyac, 2007; Javidi, 2007; Albeanu, 2008; Dehghan & Hajarian, 2010; Noor, Noor, & Waseem, 2012; Kang, Rafiq, & Kwun, 2013; Kang, Nazeer, Tanveer, Mehmood, & Rehman, 2015; McDougall, Wotherspoon, & Barker, 2019; Vanhille, 2020). Some iterative schemes need to express the derivatives to determine the zeros of the function and some do not.

The Secant method approximates the slope of the tangent at a point of the graph of the function by using the slope given by the straight passing by this point and the previous point evaluated by the algorithm. This fact, when the distance between these two points is large in relation to the curvature of the graph, reduces the convergence speed and increases the number of iterations required to fully verify the stop criterion imposed by the tolerance set by the user.

The algorithm developed by Viète in 1664, denoted here by VM, has some common points with ours. Both start from one initial estimation of the root and depend on a step h. However, unlike our technique, VM considers variations of this parameter with iterations. Initially, VM was applied to monic polynomial equations only. Since the method adds a new digit to the approximate solution at each iteration, it is seen as a finite-difference derivative-based technique for which the variable step size h_n is large at the beginning and decreases with the progress of iterations. Thus, unlike the method proposed here, the evaluation of derivatives is not separated from the iterative process. The approximation of the derivative is not as local as with our method, at least during the first iterations of the procedure. For a more detailed description of this technique, which has had a low impact on the scientific community, see Ypma (1995). Besides, Abbasbandy, Tan, & Liao (2007) introduced a new auxiliary parameter in the homotopy analysis method, which allows them to unify different methods.

The Newton-Raphson and Secant methods became a paradigm in numerical analysis. Their current importance in sciences, and especially in applied mathematics, is obvious (Mathews & Fink, 1999; Ypma, 1995; McDougall, Wotherspoon, & Barker, 2019). Thus, any improvement is clearly of interest.

The objective of this work is to develop a technique able to increase the order of convergence of the Secant method by maintaining the finite-difference concept for the derivatives and thus the user-friendly character of the method, avoiding the need for finding the derivative of the function defining the equation. To this end, we develop a new kind of algorithms by introducing a constant parameter, similar to a discretization step for the definition of the backward finite-difference formula, which is disassociated from the sequence of roots obtained by the iterations. The convergence of the new algorithm depends on the value of this parameter. The analysis shows that this technique allows us to improve the results obtained in terms of convergence regarding the Secant method and can even match the order of convergence of the Newton-Raphson method.

Section 2 develops the new algorithm, evaluates its order of convergence, and describes its computational implementation. Section 3 tests this algorithm by solving several nonlinear equations, discusses the results and gives the conclusions of this work.

2. Method

2.1 Second-Order Iterative Method

We consider the nonlinear function $f : C \subseteq \mathbb{R} \to \mathbb{R}$ of the variable x, assumed to be sufficiently differentiable on the open interval *C*, and the simple root x_0 in *C* of the equation

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

An iterative numerical method evaluates an approximation of x_0 at each iteration *n*, x_0^n, from the preceding one at iteration *n*-1, x_0^{n-1}, which is then known, and starting from an initial estimation of the root, x_0^0 (Mathews & Fink, 1999). This procedure ends after *N* iterations by finding the approximate root x_0^N that verifies the given tolerance criterion for the first time: $\left|x_0^N - x_0^{N-1}\right| \le \varepsilon$.

The Taylor expansion of *f* is written in the following form at the point x_0^n from the point x_0^{n-1} :

$$f\left(x_0^n\right)=\sum_{i=0}^{\infty}\frac{f^{(i)}\left(x_0^{n-1}\right)}{i!}\left(x_0^n - x_0^{n-1}\right)^i , \tag{2}$$

in which the upper-index (*i*) means the derivative of order *i*. Note that in the following the first, second, and third-order derivatives of *f* will be denoted by $f\,'$, $f\,''$, and $f\,'''$. The truncation of second and higher-order terms in Eq. (2), once inserted in Eq. (1), leads to the well-known second-order Newton-Raphson iterative method (Mathews & Fink, 1999), denoted by NRO2 here:

$$x_0^n = x_0^{n-1} - \frac{f\left(x_0^{n-1}\right)}{f\,'\left(x_0^{n-1}\right)} + O\left(x_0^n - x_0^{n-1}\right)^2 . \tag{3}$$

This equation means that the root at each iteration is given by the intersection of the tangent of the function (first derivative) and the *x*-axis, i.e., by the linearization of the function.

2.2 First-Order Finite-Difference Algorithm for the Second-Order Iterative Method

The major drawback of the method described in Section 2.1, as well as for all the iterative methods for which the roots are dependent on the derivatives of the function *f*, is the analytic expression of the derivatives of *f*. The numerical approximation of the derivatives involved in these methods is an option to get rid of this drawback.

The substitution of the first derivative of *f* in the NRO2 formula, Eq. (3), by a first-order backward finite-difference formula (Smith, 1986), $O\left(x_0^{n-1} - x_0^{n-2}\right)$, between two consecutive iterative points leads to the well-known Secant algorithm (Mathews & Fink, 1999), denoted by o1NRO2 here:

$$x_0^n \approx x_0^{n-1} - f\left(x_0^{n-1}\right)\frac{x_0^{n-1} - x_0^{n-2}}{f\left(x_0^{n-1}\right) - f\left(x_0^{n-2}\right)}. \tag{4}$$

This method requires two initial estimation roots, x_0^0 and x_0^1, and starts the evaluation of the approximations from the 2nd iteration.

2.3 Disassociation of Discretization and Iterations

In o1NRO2, Eq. (4), the discretization used for the calculation of the derivatives is imposed by the iterative process. This discretization is not constant during the entire procedure, it changes from one iteration to another, and thus it is not controlled by the user. It can even lead to very bad approximations and erroneous evaluations of the derivative when both consecutive iteration points are separated by a large distance; in this case, the finite-difference approximation cannot approximate the local character of the derivative.

We propose an alternative for this method to get rid of this major drawback. It consists of the disassociation of the discretization for the finite-difference approximation from the iterative progression. This technique allows us to approximate the required derivative during the current iteration through an imposed discretization step, *h*, controlled by the user, only from the last value of the approximate root, evaluated during the precedent iteration. Thus, this *h*-based discretization makes it possible to conserve the local character of the derivative during the approximation by using two proximate points given by the discretization step *h*. It thus gets rid of an important drawback of o1NRO2, i.e., its dependence on the arbitrary difference between several values of the approximate root, evaluated during several former iterations, for the evaluation of this derivative.

We apply this development to the o1NRO2 method, Eq. (4), by considering the two equidistant points belonging to the same iteration, i.e., the initial estimation is set, x_0^0, and the other initial point is evaluated from this point by considering the discretization step *h* for finite differentiations, $x_1^0 = x_0^0 - h$. These two points, x_0^0 and x_1^0, are used during the first iteration to find the approximation x_0^1. After each completed iteration giving the new approximate root x_0^n, the other point is set by $x_1^n = x_0^n - h$, and these two points are used during the following iteration to find the approximation x_0^{n+1}. With this concept, an extra point is set before each iteration to evaluate the discrete derivative, $O(h)$. The formula is thus:

$$x_0^n \approx x_0^{n-1} - f\left(x_0^{n-1}\right)\frac{x_0^{n-1} - x_1^{n-1}}{f\left(x_0^{n-1}\right) - f\left(x_1^{n-1}\right)}, \quad x_0^n \approx x_0^{n-1} - f\left(x_0^{n-1}\right)\frac{h}{f\left(x_0^{n-1}\right) - f\left(x_1^{n-1}\right)}. \tag{5}$$

This method is named ho1NRO2.

2.4 Order of Convergence

The convergence of NRO2, Eq. (3), is known to be of order 2. The convergence of o1NRO2, Eq. (4), is known to be of order 1.618 (golden ratio) (Mathews and Fink, 1999). Now the order of convergence of the disassociated technique ho1NRO2, Eq. (5), is analyzed.

The development in Section 2.3 gives rise to a two-parameter dependence of the method, on $\left(x_0^n - x_0^{n-1}\right)$ and *h*, for which the approximation is respectively $O\left(x_0^n - x_0^{n-1}\right)^2$ and $O(h)$. Thus, the convergence rate of ho1NRO2 depends on two different concepts, the progression of $\left|x_0^n - x_0^{n-1}\right|$ throughout iterations n and the definition of the derivatives at each iteration of the numerical scheme using the new parameter *h*. It is thus expected that the order of convergence of the algorithm will be *h*-dependent as well.

Definition 1. Considering the convergent sequence $\left\{x_0^n\right\}$ to x_0, if the real positive constant $K > 0$ and the exponent $k > 0$ exist such that

$$\lim_{n\to\infty}\frac{\left|x_0^{n+1} - x_0\right|}{\left|x_0^n - x_0\right|^k} = K, \tag{6}$$

then *k* is the order of convergence of the sequence and *K* is the asymptotic error constant.

Theorem 1. Let f be a sufficiently differentiable real function of variable *x*, at least $C^3(C,\mathbb{R})$, defined on the open interval *C*, $f: C \subseteq \mathbb{R} \to \mathbb{R}$, and $x_0 \in C$ be the simple root of the nonlinear equation $f(x) = 0$, then the convergence of the *h*-dependent ho1NRO2 algorithm, Eq. (5), is of order 1 (linear convergence) for finite *h* values and order 2 (quadratic convergence) for infinitesimal *h* values.

Proof of Theorem 1. We define the error made at the nth iteration by using the ho1NRO2 formula between the approximation obtained x_0^n and the root of the nonlinear equation x_0, $E_n = x_0^n - x_0$, and the error made by using the ho1NRO2 formula between the approximation defined by the step $h = x_0^n - x_1^n$ used at the n^{th} iteration x_1^n (see Section 2.3) and the root of the nonlinear equation x_0, $E_{1n} = x_1^n - x_0 = x_0^n - h - x_0 = E_n - h$. The Taylor expansion of f at x_0^n around x_0 is

$$\begin{aligned} f\left(x_0^n\right) &= f\left(x_0\right) + \left(x_0^n - x_0\right) f'\left(x_0\right) + \frac{\left(x_0^n - x_0\right)^2}{2} f''\left(x_0\right) + \frac{\left(x_0^n - x_0\right)^3}{6} f'''\left(x_0\right) + \cdots \\ &= 0 + f'\left(x_0\right)\left(E_n + \frac{E_n^{\ 2}}{2}\frac{f''\left(x_0\right)}{f'\left(x_0\right)} + \frac{E_n^{\ 3}}{6}\frac{f'''\left(x_0\right)}{f'\left(x_0\right)} + \cdots\right), \end{aligned}$$

note that by definition $f\left(x_0\right) = 0$, and the Taylor expansion of f at x_1^n around x_0 is

$$\begin{aligned} f\left(x_1^n\right) &= f\left(x_0\right) + \left(x_1^n - x_0\right) f'\left(x_0\right) + \frac{\left(x_1^n - x_0\right)^2}{2} f''\left(x_0\right) + \frac{\left(x_1^n - x_0\right)^3}{6} f'''\left(x_0\right) + \cdots \\ &= 0 + f'\left(x_0\right)\left(E_{1n} + \frac{E_{1n}^{\ 2}}{2}\frac{f''\left(x_0\right)}{f'\left(x_0\right)} + \frac{E_{1n}^{\ 3}}{6}\frac{f'''\left(x_0\right)}{f'\left(x_0\right)} + \cdots\right). \end{aligned}$$

After subtracting x_0 from both of its sides the ho1NRO2 scheme, Eq. (5), written at x_0^{n+1} is thus

$$E_{n+1} = E_n - h\frac{f'\left(x_0\right)\left(E_n + \frac{E_n^{\ 2}}{2}\frac{f''\left(x_0\right)}{f'\left(x_0\right)} + \frac{E_n^{\ 3}}{6}\frac{f'''\left(x_0\right)}{f'\left(x_0\right)} + \cdots\right)}{f'\left(x_0\right)\left(E_n + \frac{E_n^{\ 2}}{2}\frac{f''\left(x_0\right)}{f'\left(x_0\right)} + \frac{E_n^{\ 3}}{6}\frac{f'''\left(x_0\right)}{f'\left(x_0\right)} + \cdots\right) - f'\left(x_0\right)\left(E_{1n} + \frac{E_{1n}^{\ 2}}{2}\frac{f''\left(x_0\right)}{f'\left(x_0\right)} + \frac{E_{1n}^{\ 3}}{6}\frac{f'''\left(x_0\right)}{f'\left(x_0\right)} + \cdots\right)},$$

and after defining the constants $c_2 = f''\left(x_0\right)/f'\left(x_0\right)$, $c_3 = f'''\left(x_0\right)/f'\left(x_0\right)$, …, the above expression becomes

$$E_{n+1} = \frac{E_n^2 + \frac{E_n^3}{2}c_2 + \frac{E_n^4}{6}c_3 - E_nE_{1n} - \frac{E_nE_{1n}^2}{2}c_2 - \frac{E_nE_{1n}^3}{6}c_3 - hE_n - h\frac{E_n^{\ 2}}{2}c_2 - h\frac{E_n^{\ 3}}{6}c_3 + \cdots}{E_n + \frac{E_n^{\ 2}}{2}c_2 + \frac{E_n^{\ 3}}{6}c_3 - E_{1n} - \frac{E_{1n}^{\ 2}}{2}c_2 - \frac{E_{1n}^{\ 3}}{6}c_3 + \cdots}.$$

Since $E_{1n} = E_n - h$, and neglecting all the terms of third and higher degree in E_n, the above expression yields

$$E_{n+1} \approx \frac{-\frac{h^2c_2}{2}E_n + \frac{hc_2}{2}E_n^2}{\left(h - \frac{h^2c_2}{2} + \frac{h^3c_3}{6}\right) + \left(hc_2 - \frac{h^2c_3}{2}\right)E_n + \frac{hc_3}{2}E_n^2},$$

which finally reduces to

$$E_{n+1} \approx \frac{-\frac{hc_2}{2}E_n + \frac{c_2}{2}E_n^2}{1 - \frac{hc_2}{2} + \frac{h^2c_3}{6} + \left(c_2 - \frac{hc_3}{2}\right)E_n + \frac{c_3}{2}E_n^2}, \tag{7}$$

and thus

$$E_{n+1} \approx E_n\frac{-\frac{hc_2}{2} + \frac{c_2}{2}E_n}{1 - \frac{hc_2}{2} + \frac{h^2c_3}{6} + \left(c_2 - \frac{hc_3}{2}\right)E_n + \frac{c_3}{2}E_n^2}.$$

Thus, by assuming the convergence of the sequence, i.e., $E_n \xrightarrow[n\to\infty]{} 0$, the last relation leads to

$$\lim_{n\to\infty}\frac{\left|E_{n+1}\right|}{\left|E_n\right|^1} = \left|\frac{-\frac{hc_2}{2}}{1 - \frac{hc_2}{2} + \frac{h^2c_3}{6}}\right| = \left|\frac{-3hc_2}{6 - 3hc_2 + h^2c_3}\right|. \tag{8}$$

Following Def. 1, Eq. (6), this proves that

$$k = 1 \quad \text{and} \quad K = \left|\frac{-3hc_2}{6 - 3hc_2 + h^2c_3}\right|, \tag{9}$$

i.e., the order of convergence of ho1NRO2, Eq. (5), is 1.

However, this result is *h*-dependent and holds for finite values of the parameter *h*. In case $h \to 0$, Eq. (7) is written

$$E_{n+1} \approx \frac{\frac{c_2}{2} E_n^2}{1 + c_2 E_n + \frac{c_3}{2} E_n^2}, \tag{10}$$

and we have

$$E_{n+1} \approx E_n^2 \frac{\frac{c_2}{2}}{1 + c_2 E_n + \frac{c_3}{2} E_n^2}.$$

Since $E_n \xrightarrow[n\to\infty]{} 0$ by assuming the convergence of the sequence, the last relation gives us:

$$\lim_{n\to\infty} \frac{|E_{n+1}|}{|E_n|^2} = \left|\frac{c_2}{2}\right|, \tag{11}$$

which, by Def. 1, Eq. (6), proves that

$$k = 2 \quad \text{and} \quad K = \left|\frac{c_2}{2}\right|, \tag{12}$$

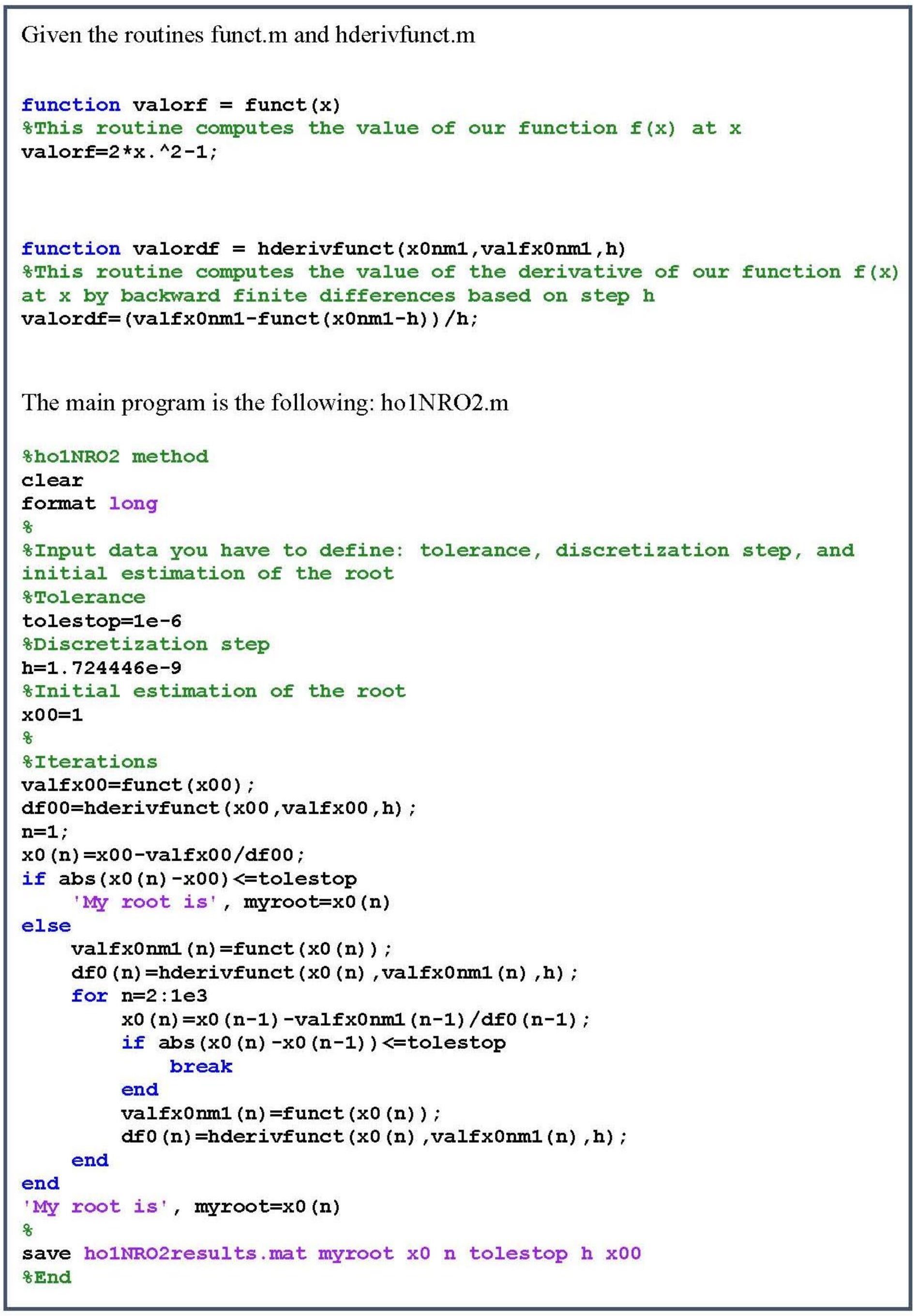
i.e., the order of convergence of ho1NRO2, Eq. (5), is 2 for infinitesimal values of *h*. □

Theorem 1 states that the parameter *h* introduced in the algorithm makes it possible when it tends to zero to obtain the same order of convergence as NRO2, although the derivative of the function defining the nonlinear equation is not evaluated exactly, but by an *h*-dependent numerical approximation. The new *h*-dependent ho1NRO2 algorithm possesses the benefits of both NRO2 and o1NRO2, i.e., the convergence speed of NRO2 and the fact that it avoids the exact expression of the derivative of the function defining the nonlinear equation like o1NRO2.

Moreover, it will be seen in Section 3 that a transient range of *h* values exists, for which the order of convergence of ho1NRO2 is between 1 and 2.

2.5 Implementation

The implementation of the new algorithm ho1NRO2 in the Matlab® R2017a environment is represented in Fig. 1.



Given the routines funct.m and hderivfunct.m

```
function valorf = funct(x)
%This routine computes the value of our function f(x) at x
valorf=2*x.^2-1;



function valordf = hderivfunct(x0nm1,valfx0nm1,h)
%This routine computes the value of the derivative of our function f(x)
at x by backward finite differences based on step h
valordf=(valfx0nm1-funct(x0nm1-h))/h;
```

The main program is the following: ho1NRO2.m

```
%ho1NRO2 method
clear
format long
%
%Input data you have to define: tolerance, discretization step, and
initial estimation of the root
%Tolerance
tolestop=1e-6
%Discretization step
h=1.724446e-9
%Initial estimation of the root
x00=1
%
%Iterations
valfx00=funct(x00);
df00=hderivfunct(x00,valfx00,h);
n=1;
x0(n)=x00-valfx00/df00;
if abs(x0(n)-x00)<=tolestop
    'My root is', myroot=x0(n)
else
    valfx0nm1(n)=funct(x0(n));
    df0(n)=hderivfunct(x0(n),valfx0nm1(n),h);
    for n=2:1e3
        x0(n)=x0(n-1)-valfx0nm1(n-1)/df0(n-1);
        if abs(x0(n)-x0(n-1))<=tolestop
            break
        end
        valfx0nm1(n)=funct(x0(n));
        df0(n)=hderivfunct(x0(n),valfx0nm1(n),h);
    end
end
'My root is', myroot=x0(n)
%
save ho1NRO2results.mat myroot x0 n tolestop h x00
%End
```

Figure 1. Implementation of the ho1NRO2 algorithm on Matlab®

3. Results and Discussion

The new h-dependent iterative algorithm developed in Section 2, ho1NRO2, is compared to the classic methods NRO2, o1NRO2, and VM by solving several nonlinear equations defined by quadratic, cubic, exponential, and logarithmic functions. Their results and performance in terms of speed of convergence are analyzed. The graphical representations of these functions and their first derivatives are displayed in Fig. 2, which shows the roots we approximate in the following Sections.

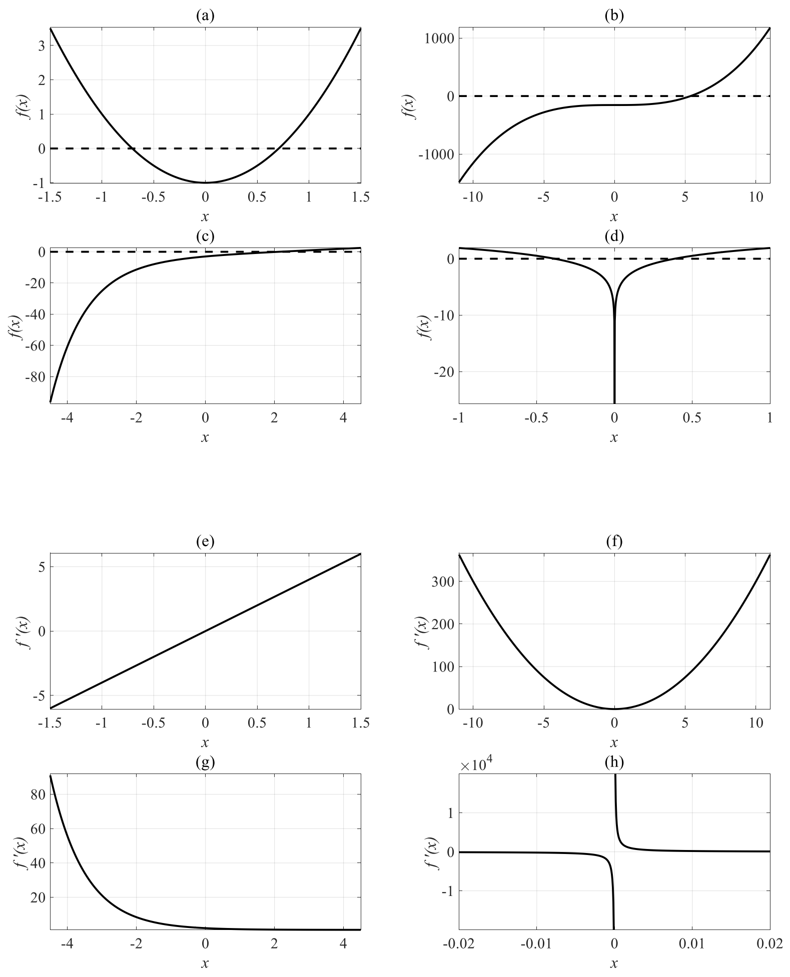


Figure 2. Functions analyzed in Sections 3.1 to 3.4. Quadratic function (a) and its first derivative (e); cubic function (b) and its first derivative (f); exponential function (c) and its first derivative (g); logarithmic function (d) and its first derivative (h)

In Sections 3.1 to 3.4 the tolerance is set at $\varepsilon = 1\times10^{-6}$ for all the cases. Tables 1-4 give the results obtained by applying the different methods. $h = 1.724446\times10^{-9}$ is used for ho1NRO2 (see Section 3.5). In these tables, bold numbers indicate the initial estimation point used for each algorithm (the entire number just above the root we seek). The corresponding percentual relative errors ($E_{r\%n} = 100|E_n|/|x_0|$) are given in Tables 1bis, 2bis, 3bis, and 4bis.

3.1 Quadratic Function

We consider $f(x) = 2x^2 - 1$ and we aim to approximate its root x_0 close to 0.7071 (see Fig. 2a). Table 1 gives the results.

Table 1. Quadratic function. x_0^N

	NRO2 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-3} to 10^{-7}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	1	**1**	**1**	**1**
n=1	0.750000000000000	**1.0010**	0.749874937468731	0.750000004409467
2	0.708333333333333	0.750124937531224	0.708323624135428	0.708333334962431
3	0.707107843137255	0.714326565546070	0.707107817823362	0.707107843135664
4	0.707106781187345	0.707318861826774	0.707106781186574	0.707106781187376
5	0.707106781186548	0.707107858231822	0.707106781186547	0.707106781186547
6		0.707106781348041		
7		0.707106781186548		
N	5	7	5	5

Table 1bis. Quadratic function. $E_{r\%n}$

	NRO2 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-3} to 10^{-7}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	(%)	(%)	(%)	(%)
n=1	6.066017177982122	41.562777593546791	6.048330665195590	6.066017801574880
2	0.173460668094231	6.083686013092777	0.172087580158467	0.173460898483444
3	0.000150182509294	1.021031695864544	0.000146602584216	0.000150182284362
4	0.000000000112764	0.029992731772444	0.000000000003800	0.000000000117207
5	0	0.000152317203405	0.000000000000016	0.000000000000016
6		0.000000022838596		
7		0.000000000000016		
N	5	7	5	5

3.2 Cubic Function

We consider $f(x) = x^3 - 155$ and we aim to approximate its root x_0 close to 5.3717 (see Fig. 2b). Table 2 gives the results.

Table 2. Cubic function. x_0^N

	NR02 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-2} to 10^{-5}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	6	**6**	**6**	**6**
n=1	5.435185185185185	**6.0010**	5.434242780335010	5.435185325786830
2	5.372424340889777	5.435279310528978	5.372391353412576	5.372424360539804
3	5.371685456588923	5.378537301040813	5.371685434578001	5.371685456634618
4	5.371685354944834	5.371765769136402	5.371685354944686	5.371685354944860
5		5.371685457430352		
6		5.371685354946367		
7				
N	4	6	4	4

Table 2bis. Cubic function. $E_{r\%n}$

	NR02 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-2} to 10^{-5}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	(%)	(%)	(%)	(%)
n=1	1.182121178819590	11.715404076596952	1.164577246368159	1.182123796278277
2	0.013757059397833	1.183873428580576	0.013142960190215	0.013757425205324
3	0.000001892219753	0.127556728349210	0.000001482461523	0.000001893070417
4	0.000000000000033	0.001497001150589	0.000000000002728	0.000000000000513
5		0.000001907883892		
6		0.000000000028572		
7				
N	4	6	4	4

3.3 Exponential Function

We consider $f(x)=x-2-e^{-x}$ and we aim to approximate its root x_0 close to 2.1200 (see Fig. 2c). Table 3 gives the results.

Table 3. Exponential function. x_0^N

	NR02 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-2} to 10^{-5}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	3	**3**	**3**	**3**
n=1	2.094851746355134	**3.0010**	2.095067049610862	2.094851701005775
2	2.119993793931483	2.094830289276154	2.119993018500546	2.119993794916702
3	2.120028238924066	2.120974754942487	2.120028238732440	2.120028238925821
4	2.120028238987641	2.120029525658842	2.120028238987641	2.120028238987641
5		2.120028238922403		
6		2.120028238987641		
N	4	6	4	4

Table 3bis. Exponential function. $E_{r\%n}$

	NR02 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-2} to 10^{-5}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	(%)	(%)	(%)	(%)
n=1	1.187554588637464	41.554718225500686	1.177398909964446	1.187556727729676
2	0.001624745157851	1.188566701522813	0.001661321601665	0.001624698685895
3	0.000000002998777	0.044646384299927	0.000000012037653	0.000000002916014
4	0	0.000060691229345	0	0
5		0.000000003077246		
6		0		
N	4	6	4	4

3.4 Logarithmic Function

We consider $f(x) = \ln(x^2/2) + 2.6$ and we aim to approximate its root x_0 close to 0.3854 (see Fig. 2d). Table 4 gives the results.

Table 4. Logarithmic function. x_0^N

	NRO2 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-3} to 10^{-9}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	1	**1**	**1**	**1**
n=1	0.046573590279973	**1.0010**	0.047050382976709	0.046573545713422
2	0.144997336042209	0.046096956487572	0.145897314713615	0.144997239243819
3	0.286748689095396	0.704904521901549	0.287620875947528	0.286748596694532
4	0.371546781313959	0.559067268009040	0.371803235317036	0.371546756693845
5	0.385165489567856	0.325052664895034	0.385174799373750	0.385165489131901
6	0.385418075047655	0.398561266403188	0.385418081035310	0.385418075051371
7	0.385418157886549	0.386470873817560	0.385418157886546	0.385418157886550
8		0.385400317074306		
9		0.385418182240515		
10		0.385418157887122		
N	7	10	7	7

Table 4bis. Logarithmic function. $E_{r\%n}$

	NRO2 Eq. (3)	o1NRO2 Eq. (4)	VM Ref. (Ypma, 1995)	ho1NRO2 Eq. (5)
h			variable from 10^{-3} to 10^{-9}	1.724446×10^{-9}
ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
	(%)	(%)	(%)	(%)
n=1	87.916088194868891	159.7179140414627	87.792380298139065	87.916099758037362
2	62.379215126422153	88.0397548625252	62.145708050278834	62.379240241582721
3	25.600627986033732	82.8934385880769	25.374331732397287	25.600651960219203
4	3.599045942376669	45.0547299262410	3.532506782809423	3.599052330273382
5	0.065556931745892	15.6623375822610	0.063141423886808	0.065557044858159
6	0.000021493253795	3.4100906373224	0.000019939706023	0.000021492289799
7	0.000000000002304	0.2731360496283	0.000000000003082	0.000000000002132
8		0.0046289495932		
9		0.0000063188399		
10		0.0000000001462		
N	7	10	7	7

In all the cases shown above ho1NRO2 converges to the solution faster than o1NRO2 and as fast as NRO2. This means that what is lost with o1NRO2 by approximating the derivative of the function from NRO2 is now compensated for with the introduction of the new parameter *h* in ho1NRO2. Note that VM behaves similarly to NRO2 and ho1NRO2. However, regarding the new algorithm proposed in this paper, ho1NRO2, VM requires the definition of the highest exponent of 10 in the sought root (see Section 1), which is not always easy to find, especially in cases for which automatization is required by repeating the process in a subroutine for different functions within a global code.

It can be seen in Table 1 for the quadratic function that the introduction of the parameter *h* modifies the application of the derivative from o1NRO2 to ho1NRO2. For example, at the fourth iteration, $n = 4$, o1NRO2 computes the derivative between the points 0.714326565546070 and 0.750124937531224, giving the value 2.928903006154588, whereas ho1NRO2 computes the derivative between the points 0.707107843135664 and 0.707107841411218 (with *h*=1.724446×10^{-9}), giving the value 2.828431456205209, which is a much more local approximation of the derivative. Note that NRO2 evaluates the derivatives locally, from the analytic expression of the derivative function, which is at the fourth iteration the derivative value at the point 0.707107843137255, equal to 2.82843137254902. The sequences of derivatives used at each iteration are the following, respectively for NRO2, o1NRO2, and ho1NRO2: {4, 3, 2.833333333333334, 2.828431372549020, 2.828427124749380}; {not evaluated, 4.001999999999836, 3.502249875062447, 2.928903006154588, 2.843290854745681, 2.828853440117567, 2.828429279234783}; {4.000000070551469, 3.000000117295034, 2.833333329656097, 2.828431456205209, 2.828427207030713}. This shows that ho1NRO2 uses derivative values that are much closer to the ones used by NRO2 than o1NRO2.

The new *h*-dependent method, ho1NRO2, is now tested by varying the initial estimation point x_0^0 and the discretization step *h* (*h*-convergence, ranging from 1×10^{-9} to 1×10^{-1}) in the case of the quadratic function. The results are displayed in Fig. 3.

As expected, the number of iterations required to hold the stop criterion depends on the estimation point x_0^0. Also, a clear dependence on the step value *h* is observed, even if a good approximation is obtained whatever the step value *h* is used (Fig. 3a), the convergence speed depends on the *h* value (Fig. 3b). The method requires fewer iterations as *h* becomes smaller. For x_0^0 very close to x_0 and a very small *h* value, ho1NRO2 only needs 3 iterations to match the tolerance and reach the precision imposed by the user.

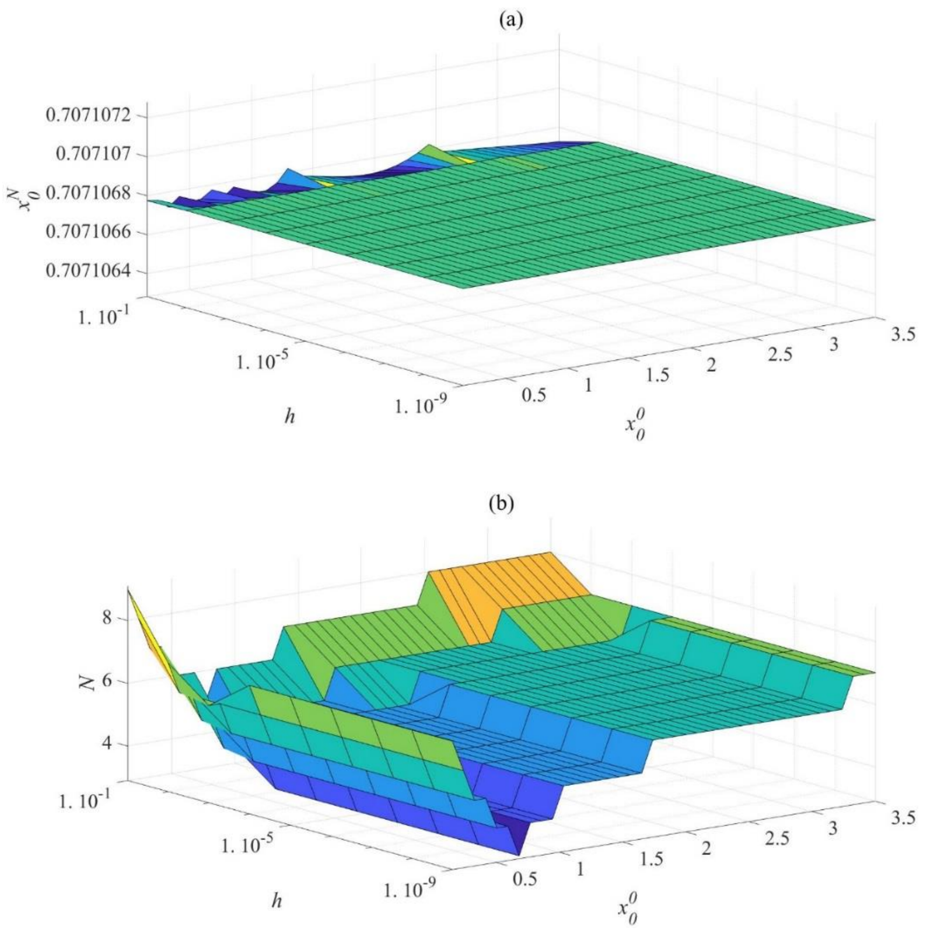
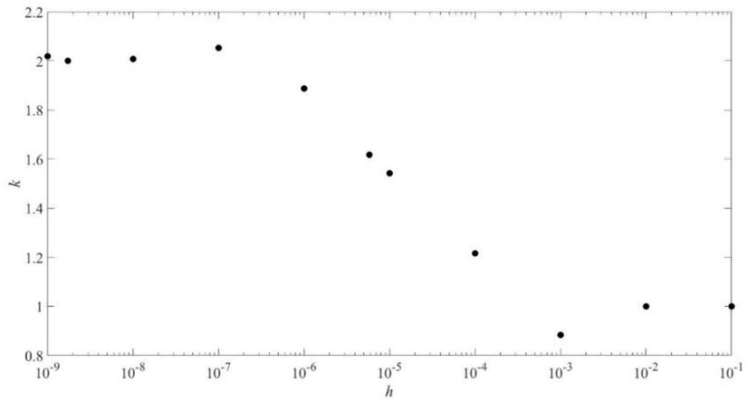


Figure 3. Quadratic function. Number of iterations required *N* (b) for the evaluation of x_0^N (a) obtained with the ho1NRO2 algorithm vs. *h* (in logarithmic scale) for a large range of x_0^0 values. ε=1×10^{-6}

3.5 Order of Convergence of ho1NRO2

To illustrate the results of Theorem 1, we now evaluate the order of convergence of h01NRO2 for several values of the parameter *h*. To this end, we consider the quadratic equation of Section 3.1.1: $f(x)=2x^2-1$ with the tolerance set at $\varepsilon=1\times10^{-6}$ and the initial estimation defined at $x_0^0=3$ to approximate the root x_0 close to 0.7071 (see Fig. 1a). Note that the initial estimation value is chosen far from the exact root to make the *N* value not too short to see the evolution of the convergence. Following Def. 1, Eq. (6), for each *h* value, we aim to find the *k* value for which the quantity $|E_{n+1}|/|E_n|^k$ tends to a constant, which is thus the order of convergence of the iterative scheme. Figure 4 represents the result of this study, for the entire variation of *h* (in logarithmic scale) (top) and the lower interval of variation of *h* (in linear scale) (bottom). It can be seen that *k* is *h*-dependent, as seen in Theorem 1, following an increasing progression as *h* is lowered. Moreover, $k<1.618$ for $h>5.817\times10^{-6}$, which means that ho1NRO2 has a lower order of convergence than o1NRO2, and $k > 1.618$ for $h < 5.817\times10^{-6}$, which means that ho1NRO2 has a higher order of convergence than o1NRO2. The order of convergence of ho1NRO2 is as high as NRO2, $k = 2$, for the smallest values of *h*, $h \le 1\times10^{-7}$. These considerations match the conclusions of Theorem 1. They are very useful since one can control the behavior of the new method by changing the *h* value. For infinitesimal values of *h*, the algorithm behaves as good as NRO2 does, but with the advantage of avoiding the analytic evaluation of the derivative of the function defining the nonlinear equation, which is a very useful computational advantage.



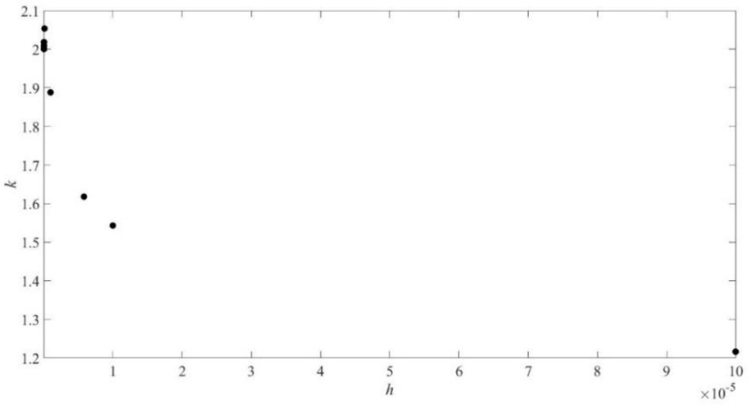


Figure 4. Order of convergence *k* of ho1NRO2 algorithm vs. *h*, for all *h* values in logarithmic scale (top) and up to 1×10^{-4} in linear scale (bottom)

Tables 5-7 show in detail the order of convergence for three *h* values: $h = 1\times10^{-1}$, $h = 5.817\times10^{-6}$, and $h = 1.724446\times10^{-9}$, respectively. We compute E_n and $|E_{n+1}|/|E_n|^k$ for $k = 1$, $k = 1.618$ (the order for convergence of o1NRO2), and $k = 2$ (the order for convergence of NRO2). Figures 5-7 represent the sequence of error, $|E_{n+1}|/|E_n|^k$, for the three *h* values $h = 1\times10^{-1}$, $h = 5.817\times10^{-6}$, and $h = 1.724446\times10^{-9}$, respectively, each one for the three *k* values $k = 1$, $k = 1.618$, and $k = 2$.

For $h = 1\times10^{-1}$ with $k = 1$ we found that $|E_{n+1}|/|E_n|^k$ tends to the real positive constant $K = 7.6091\times10^{-2}$, which is equal to the theoretical constant obtained in Theorem 1 for finite *h* values, Eq. (9), $K = \left|-3hc_2/\left(6-3hc_2+h^2c_3\right)\right| = 7.6091\times10^{-2}$. For $h = 5.817\times10^{-6}$ with $k = 1.618$ we found that $|E_{n+1}|/|E_n|^k$ tends to the real positive constant $K = 4.2405\times10^{-2}$. For $h = 1.724446\times10^{-9}$ with $k = 2$ we found that $|E_{n+1}|/|E_n|^k$ tends to the real positive constant $K = 7.0689\times10^{-1}$, which is very close to the theoretical constant obtained in Theorem 1 for infinitesimal *h* values, Eq. (12), $K = |c_2/2| = 7.0711\times10^{-1}$. For the latest value of *h*, $h = 1.724446\times10^{-9}$, we also display in Fig. 8 the sequence of error, $|E_{n+1}|/|E_n|^k$, in logarithmic scale for other *k* values to clearly observe and ensure the important result obtained at this very small *h* value. The data found in this analysis thus match the results given by Theorem 1. They confirm the linear convergence of the algorithm for finite values of *h* and the quadratic convergence of the algorithm for infinitesimal values of *h*. They also suggest the existence of an order of convergence of the algorithm between 1 and 2 for intermediate *h* values.

Table 5. Order of convergence of ho1NRO2. $h = 1\times10^{-1}$

n	x_0^n	E_n	$\|E_{n+1}\|/\|E_n\|^k$ k=1	$\|E_{n+1}\|/\|E_n\|^k$ k=1.618	$\|E_{n+1}\|/\|E_n\|^k$ k=2
	3.	2.292893218813453	0.371676816748042	0.002368519983612	0.000001596962559
1	1.559322033898303	0.852215252711756	0.249189780516529	0.002750747699533	0.000002924023945
2	0.919470112962628	0.212363331776080	0.064615982827296	0.001683467763960	0.000003042709035
3	0.720828846585738	0.013722065399191	0.064306965211721	0.009106107503229	0.000046863911037
4	0.706224356804289	-0.000882424382259	0.076865803087179	0.059334115717769	0.000871075240356
5	0.707174609445354	0.000067828258806	0.076031674310969	0.286538353546292	0.011209439199730
6	0.707101624090465	-0.000005157096083	0.076095655085445	1.409592019736821	0.147555240131942
7	0.707107173619152	0.000000392432605	0.076090789816790	6.924449794573884	1.938951781698433
N=8	0.707106751326041	-0.000000029860507		(×10^2)	(×10^5)

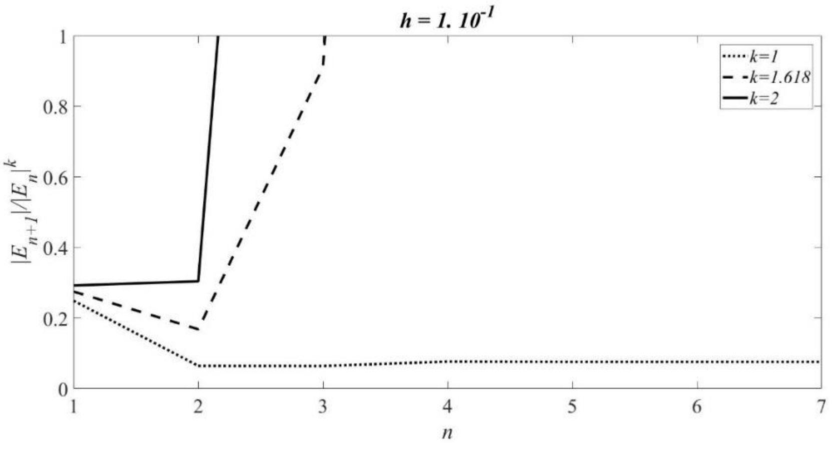


Figure 5. Sequence of error of the ho1NRO2 algorithm vs. *n* for three exponent *k* values, with *h*=1.×10^{-1}

Table 6. Order of convergence of ho1NRO2. $h = 5.817\times10^{-6}$

N	x_0^n	E_n	$\|E_{n+1}\|/\|E_n\|^k$ k=1	$\|E_{n+1}\|/\|E_n\|^k$ k=1.618	$\|E_{n+1}\|/\|E_n\|^k$ k=2
	3.	2.292893218813453	0.382147840057703	0.243524684641400	0.164195280679967
1	1.583331959860175	0.876225178673627	0.276701599357262	0.300244561209114	0.315788231258219
2	0.949559689522643	0.242452908336095	0.127663293118612	0.306454717388424	0.526548821355614
3	0.738059117890918	0.030952336704371	0.020964879981283	0.179574156008232	0.677327859977788
4	0.707755693210695	0.000648912024147	0.000454321787838	0.042406529031214	0.700128478024442
5	0.707107076001419	0.000000294814871	0.000003904790320	0.042404915025824	13.24488926558051
N=6	0.707106781185396	-0.000000000001151			

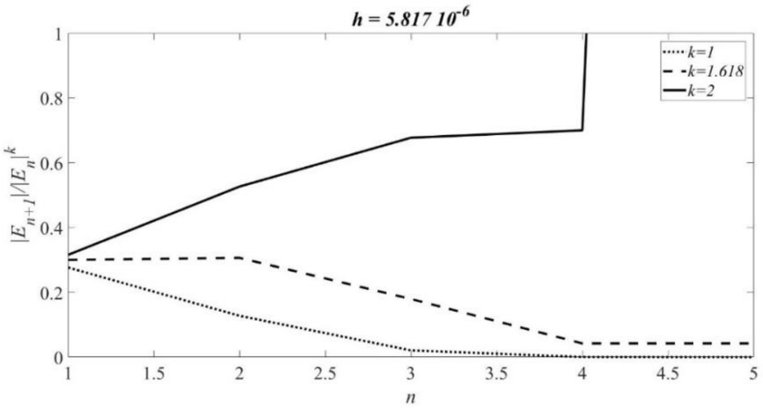


Figure 6. Sequence of error of the ho1NRO2 algorithm vs. *n* for three exponent *k* values, with *h*=5.817×10^{-6}

Table 7. Order of convergence of ho1NRO2. $h = 1.724446 \times 10^{-9}$

N	x_0^n	E_n	$\lvert E_{n+1}\rvert/\lvert E_n\rvert^k$ k=1	$\lvert E_{n+1}\rvert/\lvert E_n\rvert^k$ k=1.618	$\lvert E_{n+1}\rvert/\lvert E_n\rvert^k$ k=2
	3.	2.292893218813453	0.382148869802242	0.243525340848731	0.164195723124424
1	1.583333211723946	0.876226430537399	0.276703071901519	0.300245893945570	0.315789460644110
2	0.949561326197548	0.242454545011001	0.127666613922690	0.306461410458016	0.526558963524061
3	0.738060131978269	0.030953350791721	0.020969430651901	0.179609498051671	0.677452686560507
4	0.707755855329419	0.000649074142871	0.000458537248889	0.042793394592570	0.706448183038584
5	0.707107078811219	0.000000297624672	0.000000210387728	0.002271396780441	0.706889408870964
N=6	0.707106781186610	0.000000000000063			

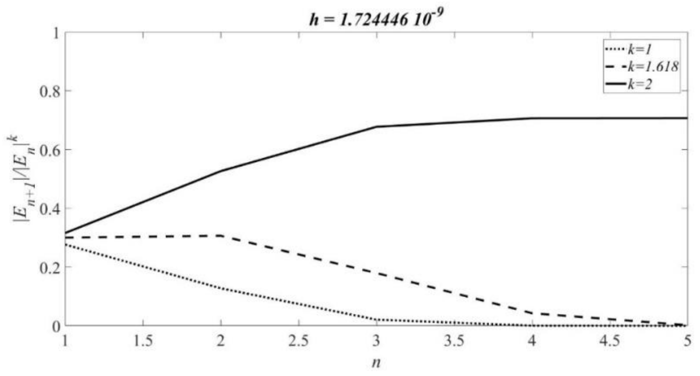


Figure 7. Sequence of error of the ho1NRO2 algorithm vs. *n* for three exponent *k* values, with *h*=1.724446 ×10^{-9}

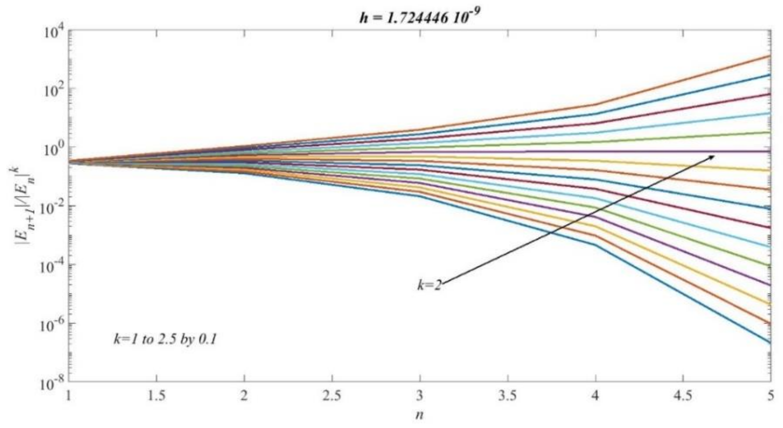


Figure 8. Sequence of error of the ho1NRO2 algorithm vs. *n* for exponent *k* varying from 1 to 2.5 by 0.1, for *h*=1.724446 ×10^{-9}

The CPU time required by ho1NRO2, carried out by running the ad-hoc functions developed on Matlab® R2017a (see Section 2.5, Fig. 1) on a 128 GB RAM computer with the 64 bits Windows 10 operative system working with an Intel® Core™ i7-6800 K CPU @ 3.40 GHz, to solve the problem of the quadratic function (Section 3.1) does not show variations for NRO2, o1NRO2, or h01NRO2; it is 0.046875 s, even by changing the initial estimation moderately. This is most likely because this CPU time is due to the prints and other interactions with the screen, which means that the CPU time due to the computations for each algorithm is negligible.

The whole resolution procedure needs $m_{No} = 2N$ operations and $m_{Nf} = 2N$ evaluations of functions or derivatives with NRO2 ($m_o = 2$ operations and $m_f = 2$ evaluations per iteration), $m_{No} = 4N - 3$ operations and $m_{Nf} = N - 1$ evaluations of functions with o1NRO2 ($m_o = 4$ operations and $m_f = 1$ evaluation per iteration), and $m_{No} = 4N$ operations and $m_{Nf} = 2N$ evaluations of functions with ho1NRO2 ($m_o = 4$ operations and $m_f = 2$ evaluations per iteration), all of $O(N)$. These numbers are respectively $m_{No} = 10$ and $m_{Nf} = 10$ for NRO2, $m_{No} = 25$ and $m_{Nf} = 6$ for o1NRO2, and $m_{No} = 20$ and $m_{Nf} = 10$ for ho1NRO2 in the case of the solution of the problem shown in Table 1 for the quadratic function (Section 3.1). VM requires the same number of evaluations but 3 more operations per iteration than ho1NRO2, because of the evaluation of h_n.

The results presented in the above section allows us to observe that the new iterative technique ho1NRO2 for finding the roots of nonlinear functions developed and implemented in this paper used with infinitesimal values of the new parameter *h* (discretization step for the approximation of the derivatives of the function) has a much better behavior than o1NRO2 (Secant method); it accelerates the convergence of the Secant method. This good behavior is the one of NRO2 (Newton-Raphson method). This improvement is due to the disassociation procedure made from the iterative progression and used in ho1NRO2 that allows us to obtain approximate results of the derivative evaluated numerically very close to the analytic value used in NRO2, whereas when no disassociation technique is employed, like with o1NRO2, the distance between two points used in this approximation can be large, causing bad approximations and lower convergence speed (ho1NRO2 with *h* tending to zero ensures an infinitesimal distance between the points involved in the evaluation of the derivative whatever the two consecutive iterative approximations are distant the one from the other, which always offers the user to calculate the approximation of the derivative locally with two proximate points, and not globally as when no disassociation is applied). Thus, by applying the disassociation procedure, the numerical derivative used in o1NRO2 tends to the analytic derivative used in NRO2, and ho1NRO2 tends to NRO2. The analysis of the results vs. *h* shows that for infinitesimal values of *h* the order of convergence of ho1NRO2 is higher than for o1NRO2 and reaches the one of NRO2. But the new technique h01NRO2 is not only able to skip up the loss in convergence speed produced when NRO2 is substituted by o1NRO2. Besides this important feature, it must be stressed that another advantage of ho1NRO2 is that it does not need any analytic evaluation of the first derivative of function *f*, whereas NRO2 requires the analytic evaluation of this derivative, whether the analytic deduction of the derivative is straightforward or not. The introduction of the new degree of freedom *h* makes the new algorithm very useful.

Theorem 1 proves that the order of convergence *k* of the algorithm depends on *h* and that, whereas it is only 1 for finite *h* values, for infinitesimal values of *h* it is equal to 2, like the Newton-Raphson method, higher than the Secant algorithm. Moreover, ho1NRO2 presents the advantage of requiring only one initial estimation, like the Newton-Raphson algorithm and unlike the Secant method, and avoids the analytic expression of the derivative of the nonlinear function, like the Secant algorithm and unlike the Newton-Raphson method, which makes the evaluation of the derivatives much easier. Thus, this new algorithm gets rid of the shortcoming of the Newton-Raphson method (the evaluation of the derivatives) and the Secant method (its order of convergence). Moreover, its implementation is straightforward. These advantages make it easy to use in many cases and allow us to obtain the approximate root quickly with huge precision. The introduction of the new parameter *h* into the iterative method provides the user with control that ensures that the approximate derivatives tend to their real local values. The test of the new algorithm by solving several nonlinear equations shows a huge concordance between the results obtained and the theoretical aspects given in Theorem 1.

These conclusions suggest studying the use of similar *h*-disassociation techniques in other existing iterative schemes in the future.

Acknowledgments

This work was supported by the Agencia Estatal de Investigación, Ministerio de Ciencia e Innovación (National Agency for Research, Ministry of Science and Innovation, Spain) and the European Regional Development Fund (FEDER) [grant number DPI2017-84758-P].

The author dedicates this work to Dr. Cleofé Campos-Pozuelo.

References

Abbasbandy, S. (2006). Modified homotopy perturbation method for nonlinear equations and comparison with Adomian decomposition method. *Appl. Math. Comput.*, *172*, 431-438. https://doi.org/10.1016/j.amc.2005.02.015

Abbasbandy, S., Tan, Y., & Liao, S. J. (2007). Newton homotopy analysis method for nonlinear equations. Appl. Math. Comput., *188*, 1794-1800. https://doi.org/10.1016/j.amc.2006.11.136

Albeanu, G. (2008). On the generalized Halley method for solving nonlinear equations. *ROMAI J.*, *4*, 1-6.

Dahlquist, G., & Björck, A. (2008). Numerical Methods in Scientific Computing Vol. I. SIAM, Philadelphia. https://doi.org/10.1137/1.9780898717785

Dehghan, M., & Hajarian, M. (2010). Some derivative free quadratic and cubic convergence iterative formulas for solving nonlinear equations. *Comput. Appl. Math.*, *29*, 19-30. https://doi.org/10.1590/S1807-03022010000100002

Gander, W. (1985). On Halley's iteration method. *The Am. Math. Monthly*, *92*, 131-134. https://doi.org/10.1080/00029890.1985.11971554

Javidi, M. (2007). Iterative methods to nonlinear equations. *Appl. Math. Comput.*, *196*, 360-365.

https://doi.org/10.1016/j.amc.2007.03.068

Kang, S. M., Rafiq, A., & Kwun, Y. C. (2013). A new second-order iteration method for solving nonlinear equations. *Abstract and Appl. Analysis*, *2013*, 487062, 1-4. https://doi.org/10.1155/2013/487062

Kang, S. M., Nazeer, W., Tanveer, M., Mehmood, Q., & Rehman, K. (2015). Improvements in Newton-Raphson method for nonlinear equations using modified Adomian decomposition method. *Int. J. Math. Anal.*, *9*, 1919-1928. https://doi.org/10.12988/ijma.2015.54124

Mathews, J. H., & Fink, K. D. (1999). *Numerical Methods using Matlab.* (3rd ed). Prentice Hall, Upper Saddle River.

McDougall, T. J., Wotherspoon, S. J., & Barker, P. M. (2019). An accelerated version of Newton's method with convergence order 3+1. *Res. Appl. Math.*, *4*, 100078. https://doi.org/10.1016/j.rinam.2019.100078

Noor, M. A., Noor, K. I., & Waseem, M. (2012). Higher-order iterative algorithms for solving nonlinear equations. *World Appl. Sci. J.*, *16*, 1657-1663.

Pakdemirli, M., & Boyac, H. (2007). Generation of root finding algorithms via perturbation theory and some formulas. *Appl. Math. Comput.*, *184*, 783-788. https://doi.org/10.1016/j.amc.2006.05.207

Smith, G. D. (1986). *Numerical Solution of Partial Differential Equations: Finite Difference Methods* (3rd ed.). Clarendon Press, Oxford.

Vanhille, C. (2020). A note on the convergence of the irrational Halley's iterative algorithm for solving nonlinear equations. *Int. J. Comput. Math.*, *97*, 1840-1848. https://doi.org/10.1080/00207160.2019.1664736

Ypma, T. J. (1995). Historical development of the Newton-Raphson method. *SIAM Rev.*, *37*, 531-551. https://doi.org/10.1137/1037125