Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 205 (2023) 678–696
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original articles

A common framework for modified Regula Falsi methods and new methods of this kind

Julio M. Fernández-Díaz[a,*], César O. Menéndez-Pérez[b]

[a] Department of Physics, University of Oviedo, E.P. de Mieres, C/Gonzalo Gutiérrez Quirós, s/n, 33600, Oviedo, Spain
[b] Department of Mathematics, University of Oviedo, C/Independencia, 13, 33004, Oviedo, Spain

Received 29 June 2022; received in revised form 22 September 2022; accepted 17 October 2022
Available online 28 October 2022

Abstract

We show that the modified Regula Falsi methods based on a scaling factor (Scaling Factor Regula Falsi, SFRF) have a common framework, and that the scaling factor depends only on two dimensionless parameters. This common framework allows for the comparison of different methods and their possible combination, without hybridization for treating some difficult cases, such as multiple roots. With this framework we prove, for all SFRF methods, the global convergence of the successive approximations to the root, and also the convergence of the bracketing interval radius to zero, for both simple and multiple roots. We show that SFRF methods only occasionally need a small scaling factor to make the radius of the interval go to zero. This way, the accumulation of the approximations to the root near an interval limit that some improvements of Regula Falsi methods suffer from is cured. As an example, new SFRF methods are exposed. One is specific for multiple roots of known multiplicity, and greatly outperforms pure bisection. Another, developed for simple roots, compares well with other methods belonging to numerical libraries that are widely used by the scientific community, and for multiple roots the new method outperforms them. The new framework could also allow the development of other SFRF methods overperforming the previously known ones.

Keywords: Real function zeros; Modified Regula Falsi; Bracketing; Global convergence

1. Introduction

The resolution of equations of the form $f(x) = 0$ is a fundamental task in many branches of science and engineering. Despite this subject's age, it has never ceased to be of interest for numerical practitioners, because no method is perfect in all cases. Very recent examples of new developments are, for example, [8,17,18].

A set of very important methods for this problem guarantees the solution is within a certain interval $[a, b]$ of $\mathbb{R}$, called methods of global convergence. In this case it must be true that $f \in C^0([a, b])$, $f(a)f(b) < 0$ (it is said the initial conditions are "bracketing").

Another important aspect is whether or not the calculation of $f'(x)$ (and possibly other higher order derivatives) is needed. The use of the derivative increases the order of convergence only near the root, so they are interesting,

* Corresponding author.
E-mail addresses: julio@uniovi.es (J.M. Fernández-Díaz), omar@uniovi.es (C.O. Menéndez-Pérez).

https://doi.org/10.1016/j.matcom.2022.10.019

although you have to code the calculation of the derivative in the user's program. Obviously, for complicated functions methods without derivatives are preferred, even though they have a lower order of convergence.

A wide literature of methods with high convergence orders (for simple roots, usually) has been developed, sometimes needing derivatives of first order and higher, and also multipoint. As noted by Petković [25] in his conclusions, if the initial approximation to the root is bad, it is not possible to reach the theoretical convergence of the method used. Practical experiments show that multipoint methods can converge very slowly at the beginning of the process. Therefore, it is possible that methods with lower convergence order, such as bisection, will work eventually better than more advanced ones (this is typical for multiple roots).

Actually, for a generic $f(x) = 0$, the most popular numerical packages do not include any method of higher order than Muller's one [20], for cases without derivatives, and Newton–Raphson one for cases with derivatives. Also, they do not include any method suited for multiple roots (except bisection, as a slow robust solution).

Among the methods without derivatives, Regula Falsi (RF) is perhaps the oldest known, with convergence order of 1 (see, [26, pp. 339–340]). Therefore, it often performs slowly, and it is not used in practice, tough it is globally convergent because the root is always inside the current interval.

To improve convergence bracket-maintaining variants have been developed, broadly named modified Regula Falsi (MRF). Initially a "scaling factor" for some ordinates was employed as in the Illinois method [11,28], the Pegasus method [12], the method of Anderson and Björck [3], and several methods of Ford [13]. More recently, in [14] some scaling factor formulas based on a heuristic are shown.

Other MRF methods with the words "Regula Falsi" in their name (or in their development) intend to improve the slow convergence of RF by adding some calculations derived from Newton–Raphson or Steffensen's methods. We mention AC [33], STFA [31], RFNL [23], EXRF [6], CIRF [7], RB and RBP [29] (the meaning of the acronyms are exposed in the citations).

Really, the methods in the latter group are a hybrid, using the RF part to keep bracketing, necessary for global convergence. The method developed by King [16] is also a hybrid: it replaces Muller's iteration formula with another, based on the Anderson and Björck's method, with the additional control of bracketing. Of this kind is also the IRF method by Naghipoor et al. [21], that hybridizes RF with a heuristically scaled RF.

Two typical methods in many numerical packages are Van Wijngaarden–Dekker–Brent's method [5, pp. 47–60], and Alefeld–Potra–Shi's [2] one. These have been developed for high convergence rates in the simple root case, and they use bisection when convergence is not rapid enough, and always in the multiple root case.

In any case, the MRF methods have not been successful in the development of computational routines (with the exception of the Illinois and Pegasus methods at the beginning of the computer era). They have been surpassed by Van Wijngaarden–Dekker–Brent and Alefeld–Potra–Shi methods. However, as we will see, it is possible to develop new MRF methods, pairing them approximately with the hybrid ones, maintaining the advantages of global convergence and bracketing.

In this work we will propose a framework to compare and combine scaling factor MRF methods. For sake of clarity we name these methods Scaling Factor Regula Falsi (SFRF), since "modified" is a very general term, shared by a lot of methods.

We will demonstrate that scaling factors depend only on two dimensionless parameters. We will also show their forms for some of the methods cited above. Additionally, we present the conditions the scale factor must fulfil for both global convergence of the approximations to the root, and the sequence of bracketing radii.

Our pure SFRF (non-hybrid) methodology allows us to change the scaling factor as desired at every step of the iteration, if needed. For example, if in the iteration we detect that there is a multiple root, we can change the function that calculates this scaling factor to another one more suitable for multiple root cases.

As an example of the new framework application, we propose three new SFRF formulas. The first one is not specifically developed for multiple roots, but its behaviour in this case is acceptable. The second one is good for single roots and it is combined with the previous one after detecting that the multiplicity is greater than one. The third is a specific development for multiple roots of known multiplicity. In an appendix we show the pseudocode of the three new methods developed here.

Finally, as a sample, we will show a comparison against several SFRF and other methods, for a set of functions, and show that our methods perform well compared with others fully established in the numerical codes currently in use.

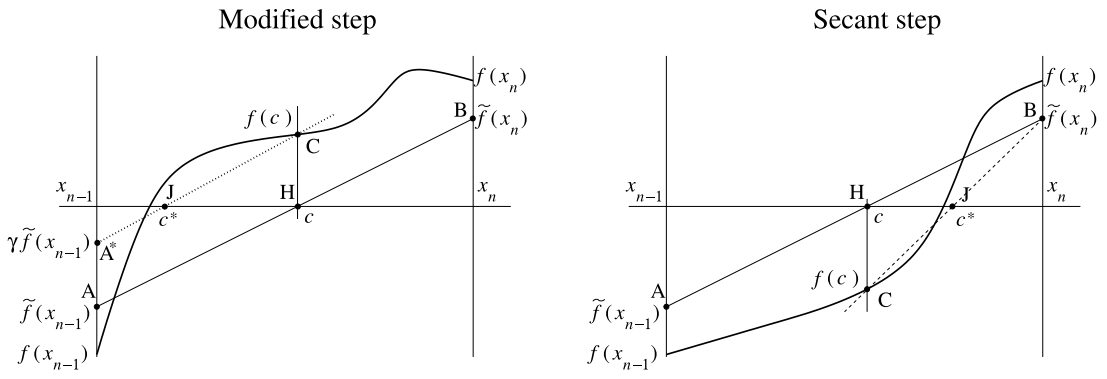

Fig. 1. Geometric description of the SFRF method. Left, for a modified step when $f(c)\tilde{f}(x_n) > 0$. By construction A, H and B are collinear. Abscissa c^* is the one obtained by SFRF method in the next step. Points C, J and A* are also collinear. Right, for a secant step, when a sign inversion, $f(c)\tilde{f}(x_n) < 0$, is produced.

2. Developing the common framework

The SFRF method can be used to calculate a root α of function $f \in C^0([a, b])$ starting with $[a, b]$ under the conditions of Bolzano's theorem, $f(a)f(b) < 0$. It generates two sequences: one, $\{x_n\}$, with the approximations to the root; another with the enclosing intervals $\{[a_n, b_n]\}$, such that:

$$x_n \in [a_n, b_n] \subset [a_{n-1}, b_{n-1}] \subset \ldots \subset [a_1, b_1] \subset [a, b], \qquad f(a_i)f(b_i) < 0, \quad i = 1, 2, \ldots, n.$$

Before describing the algorithm we need some notation.

Definition. The factor $\gamma \in \mathbb{R}$ is a scaling factor for the ordinates that fulfils $0 < \gamma < 1$, and it is in general a function of the data known at current iteration. Different SFRF methods have different ways to calculate γ.

Definition. We call an "ordinate associated to x" some value $\tilde{f}(x)$ that verifies:

$$0 < |\tilde{f}(x)| = \gamma |f(x)| \leq |f(x)|, \qquad \text{sign}(\tilde{f}(x)) = \text{sign}(f(x)). \tag{1}$$

Definition. After iteration n, we call radius of the interval $R_n = |b_n - a_n|$ where $f(a_n)f(b_n) < 0$.

Fig. 1 shows the geometrical construction of the method. Initially $x_1 = a$, $\tilde{f}(x_1) = f(a)$, $x_2 = b$, $\tilde{f}(x_2) = f(b)$ are taken. Two controlling parameters are used to limit the iterations: one is the maximum acceptable radius, *xtol*, and the other is the maximum acceptable absolute value of the function, *ftol*.

In this method we iterate the Scaling Factor Regula Falsi (SFRF) algorithm. At each step x_{n-1} and x_n are the extremes of the bracketing interval.

Algorithm 1. SFRF at step $n > 1$.
Input: $(x_{n-1}, \tilde{f}(x_{n-1}))$, $(x_n, \tilde{f}(x_n))$

(i) Let put

$$c = x_n - \tilde{f}(x_n)\frac{x_{n-1} - x_n}{\tilde{f}(x_{n-1}) - \tilde{f}(x_n)}. \tag{2}$$

(ii) (Radius control) If $|x_n - x_{n-1}| < xtol$ then c is the solution; **stop**.

(iii) (Function control) If $|f(c)| < ftol$ then c is the solution; **stop**.

(iv) (secant step) If $\tilde{f}(x_n)f(c) < 0$ (sign inversion) then **output**

$$(x_{n+1}, \tilde{f}(x_{n+1})) \leftarrow (c, f(c)),$$
$$(x_n, \tilde{f}(x_n)) \leftarrow (x_n, \tilde{f}(x_n)).$$

(v) (modified step) If $\tilde{f}(x_n)f(c) > 0$, where γ satisfies $0 < \gamma < 1$, then **output**

$$(x_{n+1}, \tilde{f}(x_{n+1})) \leftarrow (c, f(c)),$$
$$(x_n, \tilde{f}(x_n)) \leftarrow (x_{n-1}, \gamma\tilde{f}(x_{n-1})).$$

(vi) After every step n:

$$(a_{n+1}, b_{n+1}) \leftarrow (\min(x_n, x_{n+1}), \max(x_n, x_{n+1})).$$

End of algorithm SFRF at step $n > 1$.

By putting $\gamma = 1$ we recover the classic RF, and $\gamma = \frac{1}{2}$ is another used value [28]. The way in which we calculate γ from the data known at every algorithm step defines the specific SFRF method (see below).

Because the SFRF method does not require any memory apart from the extremes of the current interval and the corresponding associated ordinates, the calculation of an adequate value for γ must be obtained from the knowledge of only x_{n-1}, x_n, c, $\tilde{f}(x_{n-1})$, $\tilde{f}(x_n)$, and $f(c)$, apparently six parameters. We will demonstrate that **any SFRF method depends only on two parameters**. This fact creates a common framework that allows us to compare different SFRF methods.

Theorem 1. *Let* $(x_{n-1}, \tilde{f}(x_{n-1}))$, $(x_n, \tilde{f}(x_n))$ *the current data in step n,* $\tilde{f}(x_n) \neq 0$, $\tilde{f}(x_{n-1}) \neq 0$, $\tilde{f}(x_{n-1})\tilde{f}(x_n) < 0$, *and* $c \in (x_{n-1}, x_n)$ *obtained by (2) and known* $f(c)$, *with* $\tilde{f}(x_n)f(c) > 0$, *which implies a modified step of Algorithm 1, then the value of* γ *is only function of two dimensionless parameters, independent quotients among* $\tilde{f}(x_{n-1})$, $\tilde{f}(x_n)$ *and* $f(c)$.

Proof. By the Bolzano's theorem a root α must exist in the open interval (x_{n-1}, x_n). Let us consider the transformation:

$$(x, y) \to (z, w),$$

with:

$$z = \frac{x - x_n}{x_{n-1} - x_n}, \qquad w = \frac{y}{\tilde{f}(x_{n-1}) - \tilde{f}(x_n)}. \tag{3}$$

This converts $x \in [x_n, x_{n-1}]$ in $z \in [0, 1]$.

Then

$$g(z) = \frac{\tilde{f}(x(z))}{\tilde{f}(x_{n-1}) - \tilde{f}(x_n)}, \tag{4}$$

is the function transformed of $\tilde{f}$. If a root of $f(x)$ is in the interval, $\alpha \in (x_{n-1}, x_n)$, then by (3) we have $0 < \alpha^* < 1$, $\alpha^* = (\alpha - x_n)/(x_{n-1} - x_n)$, being in this case $g(\alpha^*) = f(\alpha) = 0$, so that the new function $g(z)$ also has a root $z \in (0, 1)$.

With transformations (3) and (4), the function values at the extremes of the current interval are:

$$g_0 = g(0) = \frac{\tilde{f}(x_n)}{\tilde{f}(x_{n-1}) - \tilde{f}(x_n)} = \frac{1}{\frac{\tilde{f}(x_{n-1})}{\tilde{f}(x_n)} - 1}, \tag{5}$$

and:

$$g_1 = g(1) = \frac{\tilde{f}(x_{n-1})}{\tilde{f}(x_{n-1}) - \tilde{f}(x_n)} = 1 + g_0. \tag{6}$$

Because $\tilde{f}(x_{n-1})/\tilde{f}(x_n) < 0$, we have $-1 < g_0 < 0$, $0 < g_1 < 1$. The transformed value of $f(c)$ is:

$$g_c = g(c) = \frac{f(c)}{\tilde{f}(x_{n-1}) - \tilde{f}(x_n)} = \frac{\frac{f(c)}{\tilde{f}(x_{n-1})}}{1 - \frac{\tilde{f}(x_n)}{\tilde{f}(x_{n-1})}}, \tag{7}$$

which has the same sign of g_0.

Besides, with (2) and (3):

$$z_c = z(c) = \frac{c - x_n}{x_{n-1} - x_n} = -\frac{\tilde{f}(x_n)}{\tilde{f}(x_{n-1}) - \tilde{f}(x_n)} = -g_0. \tag{8}$$

At the current iteration, in transformed coordinates the data we have are g_0, g_1, g_c and z_c, and, by construction, γ has to be only function of these parameters, which in (5), (6), (7) and (8) are only function of the quotients $f(c)/\tilde{f}(x_{n-1})$ and $\tilde{f}(x_n)/\tilde{f}(x_{n-1})$. Obviously, any other quotients including the values $\tilde{f}(x_{n-1})$, $\tilde{f}(x_n)$ and $f(c)$ are valid.

Without loss of generality, due to the freedom in the choice of these quotients, in the following we use:

$$\xi = \frac{f(c)}{\tilde{f}(x_n)}, \qquad \zeta = -\frac{f(c)}{\tilde{f}(x_{n-1})}, \tag{9}$$

that are always positive. Then we put in a general way:

$$\gamma = F(\xi, \zeta). \quad \square \tag{10}$$

A main fact of the SFRF methods is the collinearity of B, H and A in Fig. 1. Actually, this is the property that allows the necessity of only two control parameters. Other methods that do not satisfy this collinearity would have a similar behaviour but also the abscissae x_{n-1}, x_n and c would appear in controlling them.

F is a function that defines the specific SFRF method. The most common functions are:

1. Classic Regula Falsi: $F(\xi, \zeta) = 1$.
2. Illinois method [11,28]: $F(\xi, \zeta) = \frac{1}{2}$.
3. Pegasus method [12]: $F(\xi, \zeta) = 1/(1 + \xi)$.
4. Anderson and Björck (A&B) method [3]:

$$F(\xi, \zeta) = \begin{cases} 1 - \xi, & \xi < 1, \\ \frac{1}{2}, & \xi \geq 1. \end{cases} \tag{11}$$

5. Ford methods [13], with several formulas similar to the A&B method. For example, his fourth method has $F(\xi, \zeta) = 1 - \xi + \zeta$.

3. General convergence properties

The order of convergence, $r \geq 1$, of a succession of approximations, x_n, to the root α is defined from:

$$\lim_{n \to \infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^r} = C > 0. \tag{12}$$

On the other hand, every method of root calculation needs in each iteration step a given number of function calls, *nfc*, which usually is the most expensive part computationally. Therefore, to compare the different methods we need both the knowledge of r and *nfc*. In this respect, Traub [30, pp. 260–264] defines a computational efficiency as:

$$p = r^{1/nfc}.$$

For simple roots, the cited methods have these approximate computational efficiencies:

1. Classic Regula Falsi: $p = 1$ (linear).
2. Illinois method: $p = 1.442$.
3. Pegasus method: $p = 1.642$.
4. A&B method: $1.681 < p < 1.710$.
5. Ford fourth method: $p = 1.681$.

For multiple roots all have $p = 1$.

These values are obtained in the corresponding citations. In any case, in them, only demonstrations for convergence of x_n towards α have been provided for the simple roots case, but no demonstration about the convergence of the radii, R_n, have been shown. Also, no analysis has been performed in these citations for the multiple roots case.

We will prove the global convergence of the successive approximations, x_n, to the root, and also of the radius of the interval, R_n, to zero for all SFRF methods. Our demonstrations are valid for both simple and multiple roots. We will pose some conditions to be fulfilled by scaling factors for the convergence of all SFRF methods.

The standard notation of Taylor series around the root α is:

$$f(x_n) = \sum_{k=1}^{\infty} c_k e_n^k, \tag{13}$$

the error being

$$e_n = x_n - \alpha, \tag{14}$$

and $c_k = \frac{f^{(k)}(\alpha)}{k!}$. The sum begins at $k = 1$ because $f(\alpha) = 0$. We will use $f_n \equiv f(x_n)$ for simplifying the notation.

Our algorithm is composed of sequences of modified and secant steps. When the previous n steps are of modified kind we define a composed scaling factor:

$$\bar{\gamma}_n = \prod_{k=1}^{n} \gamma_k. \tag{15}$$

Because a secant step is really a modified one with $\gamma = 1$ after a sign inversion of $f(c)$ with respect to $f(x_n)$ we can use a more general nomenclature.

Definition. We call "Composed sequence", T_j, $j \in \mathbb{N}$, some finite set of modified steps preceded and followed by sign inversions (indicated by //):

$$T_j \equiv M_1\ M_2\ \ldots\ M_{N_{j-1}}\ M_{N_j}, \qquad N_j \in \mathbb{N}, \qquad N_j \geq 1,$$

with $\gamma_1 = 1$ (M_1 is actually a secant step).

The whole iteration process is formed by several composed sequences, in the form:

$$T_1\ //\ T_2\ //\ \ldots\ //\ T_j\ //\ \ldots\ //\ T_P,$$

with computational convergence (via radius or function control) reached in the last composed sequence, T_P, which is not exhausted.

For any error, e_n, in the sequence we call:

$$\varepsilon_n = \left|\frac{e_n}{e_0}\right|.$$

Lemma 1. *For a composed sequence of length N, which verifies $e_0 e_1 < 0$, $e_n e_{n-1} > 0$ for $1 < n < N$, it is:*

$$\frac{\varepsilon_{n+1}}{\varepsilon_n} = \frac{\bar{\gamma}_n - \bar{\Gamma}_n}{\bar{\gamma}_n + \varepsilon_n \bar{\Gamma}_n}, \tag{16}$$

$\bar{\Gamma}_n$ being a positive value.

Proof. For a composed sequence the successive points with abscissae x_n, $n > 0$, are at the same side of the root, while x_0 is at the other side, that is, $(x_n - \alpha)(x_0 - \alpha) < 0$, $n > 0$. This happens when initially a secant step is done for the next iteration, providing us f_2, such that $f_1 f_0 < 0$, $f_2 f_1 > 0$.

In this case, the successive iterations, by applying (2), satisfy:

$$x_{n+1} = \frac{\bar{\gamma}_n f_0 x_n - f_n x_0}{\bar{\gamma}_n f_0 - f_n}. \tag{17}$$

By using (14) after dividing by $e_n \neq 0$ (otherwise $x_n = \alpha$):

$$\frac{\varepsilon_{n+1}}{\varepsilon_n} = \frac{e_{n+1}}{e_n} = \frac{\bar{\gamma}_n - \frac{f_n e_0}{f_0 e_n}}{\bar{\gamma}_n + \varepsilon_n \frac{f_n e_0}{f_0 e_n}}. \tag{18}$$

Therefore

$$\bar{\Gamma}_n = \frac{f_n e_0}{f_0 e_n}, \tag{19}$$

is positive because $f_0 f_n < 0$, $e_0 e_n < 0$, due to bracketing. □

From (16) it is obvious that, for a composed sequence of only modified steps, $\bar{\Gamma}_n < \bar{\gamma}_n$ (otherwise a sign inversion would have been produced), then $0 < \bar{\Gamma}_n < 1$. Therefore we can put:

$$\bar{\Gamma}_n = \Gamma_n \bar{\gamma}_{n-1}, \qquad \Gamma_n < 1. \tag{20}$$

We need to study the successive errors generated by a composed sequence. Initially we analyse the steps before sign inversion.

Lemma 2. *Under the conditions of Lemma 1 the successive errors generated by the modified steps of a composed sequence of length N satisfy:*

$$|e_{N-1}| < |e_{N-2}| < \cdots < |e_2| < |e_1|.$$

Proof. By using Lemma 1:

$$\bar{\gamma}_n = L\bar{\Gamma}_n, \qquad 0 < n < N, \tag{21}$$

where $L > 1$ (otherwise the sequence of modified steps is broken), then

$$\frac{e_{n+1}}{e_n} = \frac{L\bar{\Gamma}_n - \bar{\Gamma}_n}{L\bar{\Gamma}_n + \varepsilon_n \bar{\Gamma}_n} = \frac{L-1}{L+\varepsilon_n}.$$

Because ε_n is positive, we obtain:

$$\frac{e_{n+1}}{e_n} < 1 - \frac{1}{L},$$

and

$$0 < \frac{e_{n+1}}{e_n} = \beta' < 1, \qquad 0 < n < N,$$

with $\beta' < 1 - 1/L$, which is equivalent to the emma statement, since the errors have the same sign. □

Now we must take the next modified step M_N that causes a sign inversion.

Lemma 3. *After the final modified step, N, of a composed sequence it is sufficient that $\gamma > \Gamma_{\min}$, for some $\Gamma_{\min} > 0$, for reducing the error of the last step approximation.*

Proof. For the last step we have:

$$\frac{e_{N+1}}{e_N} = \frac{\bar{\gamma}_N - \bar{\Gamma}_N}{\bar{\gamma}_N + \varepsilon_N \bar{\Gamma}_N} < 0, \tag{22}$$

then $\bar{\gamma}_N < \bar{\Gamma}_N$ (sign inversion), that is:

$$\bar{\gamma}_N = \frac{\bar{\Gamma}_N}{L^*}, \qquad L^* > 1.$$

Then from (22) we get:

$$\frac{e_{N+1}}{e_N} = \frac{1 - L^*}{1 + \varepsilon_N L^*},$$

and, because $e_{N+1} e_0 > 0$, $e_{N+1} e_N < 0$, equivalently

$$\frac{e_{N+1}}{e_0} = \frac{L^* - 1}{L^* + 1/\varepsilon_N}. \tag{23}$$

On the other hand:

$$L^* = \frac{\bar{\Gamma}_N}{\bar{\gamma}_N} = \frac{\Gamma_N \bar{\gamma}_{N-1}}{\gamma_N \bar{\gamma}_{N-1}} = \frac{\Gamma_N}{\gamma_N}.$$

If $\gamma_N > \Gamma_{\min} > 0$, $\Gamma_N \leq 1$ being finite:

$$L^* < \frac{\Gamma_N}{\Gamma_{\min}} = L^*_{\max}.$$

In expression (23) $L^* > 1$ and $\varepsilon_N < 1$. We want to limit the possible values of L^* because if $L^* \to \infty$ then $e_{N+1} = e_0$, which is not of interest for us (there is no radius reduction). Since $L^* < L^*_{\max} < \infty$, independently of $\varepsilon_N < 1$:

$$\frac{e_{N+1}}{e_0} < \frac{L^*_{\max} - 1}{L^*_{\max} + 1} = \beta'' < 1. \quad \square$$

The main theorem for the evolution of the interval radius throughout the calculation is:

Theorem 2. *Provided $0 < \Gamma_{\min} < \gamma < 1$, $\Gamma_{\min}$ being a small positive value, the sequence $\{R_n\}$ of the radius of the intervals $\{[a_n, b_n]\}$ with $f(a_n)f(b_n) < 0$, is convergent and:*

$$\lim_{n\to\infty} R_n = 0.$$

Since Algorithm 1 is made by composing sequences (each with possible different N_j values), we can express this theorem in a new form:

In these conditions, being $R(T_j)$ the radius of the interval after the composed sequence j, then:

$$\lim_{j\to\infty} R(T_j) = 0.$$

Proof. Without loss of generality, we assume that in the previous modified steps $a_1 = x_1$, $a_2 = x_2$, …, $a_N = x_N$, and $b_1 = b_2 = \cdots = b_N$ (that is, limit b remains the same point from the first iteration of the composed sequence). The case with interchanged roles for a and b is demonstrated in a similar way.

By Lemma 3, the right side of the interval verifies

$$\frac{e_{N+1}}{e_0} = \frac{b_{N+1} - \alpha}{b_1 - \alpha} < 1,$$

then $b_{N+1} - \alpha < b_1 - \alpha$ or

$$b_1 > b_{N+1} \tag{24}$$

For the left side of the interval, by Lemma 2, we have:

$$e_N > e_{N-1} > e_{N-2} > \cdots > e_2 > e_1,$$

since all these errors are negative. Because $a_{N+1} = a_N$ as this limit is not modified, then $a_{N+1} - \alpha = a_N - \alpha = e_N > e_1 = a_1 - \alpha$ or:

$$-a_1 > -a_{N+1}. \tag{25}$$

Now adding (24) and (25) $b_1 - a_1 > b_{N+1} - a_{N+1}$, $b_1 - a_1 > 0$ since a_1 is the inferior limit and b_1 the superior one:

$$\frac{b_{N+1} - a_{N+1}}{b_1 - a_1} < 1.$$

This fact, in notation relative to composed sequences, can be expressed as:

$$\frac{R(T_j)}{R(T_{j-1})} < 1.$$

Now taking the limit for the application of successive composed sequences:

$$\lim_{j\to\infty} R(T_j) = 0,$$

that is, a sequence of SFRF composed sequences generates a sequence of radii convergent to 0. $\square$

Theorem 3. *In SFRF methods, the approximation c to the root belongs to (x_{n-1}, x_n).*

Proof. From (2), with our new control parameters:

$$c = \beta x_n + (1-\beta)x_{n-1}, \qquad \beta = \frac{\xi}{\xi + \zeta}, \tag{26}$$

$0 < \beta < 1$ being, as $\xi > 0$ and $\zeta > 0$, then $c \in (x_{n-1}, x_n)$. □

Theorem 4. *The sequence $\{x_n\}$ generated by Algorithm 1 is convergent to a root, i.e.:*

$$\lim_{n\to\infty} x_n = \alpha. \tag{27}$$

Proof. In the application of (26), $x_{n+1} = c$. Two possibilities exist: (a) $x_{n-1} < x_n$, (b) $x_n < x_{n-1}$. For (a), since $x_{n-1} < \alpha$, $\beta \in (0, 1)$, we have:

$$\begin{aligned} |e_{n+1}| = |x_{n+1} - \alpha| &= |\beta x_n - \alpha + (1-\beta)x_{n-1}| < |\beta x_n - x_{n-1} + (1-\beta)x_{n-1}| \\ &= \beta |x_n - x_{n-1}| < |x_n - x_{n-1}|. \end{aligned}$$

For (b), because $x_n < \alpha$, $1 - \beta \in (0, 1)$, we have:

$$\begin{aligned} |e_{n+1}| = |x_{n+1} - \alpha| &= |\beta x_n - \alpha + (1-\beta)x_{n-1}| < |\beta x_n - x_n + (1-\beta)x_{n-1}| \\ &= (1-\beta)|x_{n-1} - x_n| < |x_n - x_{n-1}|. \end{aligned}$$

Hence, in both cases:

$$|e_{n+1}| < |x_n - x_{n-1}| = |b_n - a_n| = R_n.$$

Since $R_n \to 0$ by Theorem 2 then $e_{n+1} \to 0$, $x_{n+1} \to \alpha$, and (27) is accomplished. □

We have analysed the convergence to zero of e_n and R_n, but also a stopping condition related to $|f(x_n)|$ exists in the algorithm SFRF. In order to address it we need the following

Theorem 5. *Let $U(\alpha, \delta) = \{x \mid |x - \alpha| < \delta\}$ be an open domain around the root α, and a monotone function $f \in C^0(U)$ (the monotonicity is sufficient for bracketing). Then, the function values $f_n = f(x_n)$, satisfy*

$$|f_{n+1}| < |f_n|. \tag{28}$$

Proof. There are four cases to be proven: (a) $e_n > 0$ and f increasing, (b) $e_n < 0$ and f increasing, (c) $e_n > 0$ and f decreasing, and (d) $e_n < 0$ and f decreasing. We prove the first one, because the rest are proven similarly.

(a) $e_n > 0$ and f increasing. For $x \in U$, $y \in U$:

$$\frac{f(y) - f(x)}{y - x} > 0. \tag{29}$$

By Lemma 2 $|e_{n+1}| < |e_n|$, then $e_{n+1} < e_n$ and $x_{n+1} < x_n$, therefore $f(x_n) - f(x_{n+1}) > 0$. Since the root α is at left of x_n and x_{n+1}, $f_n > 0$, $f_{n+1} > 0$, then (28) is satisfied. □

Theorems 3–5 ensure that near the root the algorithm stops after a finite number of iterations because a c exists such that $|f(c)| < ftol$, and for all x such that $|x - \alpha| < |c - \alpha|$, then $|f(x)| < |f(c)|$. Also Theorem 2 ensures that after a finite number n of iterations $|R_n| < xtol$. Therefore SFRF methods are interesting, because by using adequate values $0 < \gamma < 1$ the iteration effectively converges to the desired solution, keeping bracketing. These methods do not need to hybridize.

4. Some interesting consequences of the convergence properties

We base our arguments on a sequence of modified steps. Expression (19) leads us to propose for γ_n an approximation to:

$$\gamma_n^* = \frac{f_n e_{n-1}}{f_{n-1} e_n}. \tag{30}$$

Because $\gamma_1^* = 1$, then:

$$\bar{\Gamma}_n = \prod_{k=1}^{n} \gamma_k^* = \frac{f_n e_{n-1}}{f_{n-1} e_n} \frac{f_{n-1} e_{n-2}}{f_{n-2} e_{n-1}} \cdots \frac{f_1 e_0}{f_0 e_1} = \frac{f_n e_0}{f_0 e_n},$$

and (19) is fulfilled.

For simple roots, when $f \in C^k$:

$$f_n = f(e_n) = c_1 e_n + c_k e_n^k + O(e_n^{k+1}), \quad k > 1, \tag{31}$$

$c_1 \neq 0$, $c_k \neq 0$, k being the order of the derivative above the first one giving non-zero value at the root. By using (31) in (30), in the conditions of Theorem 5, we define:

$$\gamma_n = \frac{1 + K e_n^{k-1}}{1 + K e_{n-1}^{k-1}}, \qquad K = \frac{c_k}{c_1}. \tag{32}$$

For example, for simple roots with no inflection point at the root ($k = 2$):

$$\gamma_n = \frac{1 + K e_n}{1 + K e_{n-1}}. \tag{33}$$

Except the Illinois and classical RF methods, the ones set out before are adapted to treat the simple root case, that is, they use some particular approximation compatible with (33). The formulas described for the specific cases at the end of Section 2, are developed to have a computational superlinear efficiency. Since $e_n \to 0$ in the limit $n \to \infty$, then by using (33):

$$\lim_{n\to\infty} \gamma_n = 1. \tag{34}$$

This limit has another value for multiple roots, because the formula for an adequate γ_n is not the same. In order to show this, we use a fact: according to Traub [30, p. 152], the iterations based on polynomial interpolation have linear convergence for multiple roots (except some specific methods for low m; for example, Muller's method has convergence order 1.2337 for $m = 2$, but it is linear for $m > 2$). Therefore, as all cited SFRF methods for modified steps are based on polynomial interpolation and the secant method is based on a linear one, we normally have, for iteration n in case of linear convergence:

$$e_{n+1} = C e_n + O(e_n^2), \tag{35}$$

C being a constant ($|C| < 1$) that depends on the method. Obviously, in this case, the smaller C, the better the specific method is.

For roots of multiplicity $m > 1$, by using (13) in (30), we have:

$$\gamma_n = \varepsilon_n^{m-1} + O(e_n^m), \tag{36}$$

therefore:

$$\lim_{n\to\infty} \gamma_n = C^{m-1} < 1. \tag{37}$$

The values given by the formulas for γ adapted to the simple root case, are greater than the ones that would give exactly the root in the multiple case. This explains the bad behaviour of many old SFRF methods for multiple roots.

5. New SFRF methods

At present, the search for methods for multiple roots is not over since many modern ones need large precision (over 100 decimal digits) in floating point arithmetic, so methods able to use double precision are still of interest.

In the appendix we show pseudocodes for the three new methods exposed below.

5.1. A generalized Illinois method

For multiple roots the Illinois method deserves more attention. The value of $\gamma = 1/2$ is constant but we can use other constant values. We propose here

$$F(\xi, \zeta) = G, \qquad 0 < G < 1. \tag{38}$$

We call this method the "Generalized Illinois method". With a small enough value for G from some N, $\bar{\gamma}_N = G^{N-1} \leq \varepsilon_N^{m-1}$, a secant step is forced, re-starting another composed sequence from a new e_0 in the multiple root case (and diminishing the bracketing interval from both extremes). We propose the use of a small value for G, such as:

$$\gamma = 0.1, \tag{39}$$

which numerical experiments show to be effective. We name this GIll01 (Generalized Illinois method with $\gamma = 0.1$). Further on we will present some numerical results by using this value.

5.2. A method for simple roots, but with automatic detection of multiple root cases

A method, suitable for simple roots, could be obtained by derivation of some of those previously cited. For example, A&B is a good method, but sometimes it behaves unexpectedly badly. One way to improve this is to (heuristically) modify it using the knowledge we have now. We propose:

$$\gamma = \max(1 - \xi, 0.1), \tag{40}$$

with contributions from both A&B and GIll01.

For simple roots (40) behaves reasonably well, with a typical number of function calls under $nbis/3$, $nbis$ being the number of function calls bisection would employ:

$$nbis = \left\lceil \log_2 \left(\frac{|b-a|}{xtol} \right) \right\rceil. \tag{41}$$

This factor $1/3$ has been obtained by analysing a lot of function cases with non-symmetrical (around the root) initial intervals. This behaviour is similar to those of `brentq` and `toms748` of `scipy.optimize`.

A mixture of the previous (39) and (40), named ABI01, is better than both. By counting the actual number of function calls, $nfun$, we use:

$$\gamma = \begin{cases} \max(1 - \xi, 0.1), & nfun < \frac{1}{3} nbis, \\ 0.1, & nfun \geq \frac{1}{3} nbis. \end{cases} \tag{42}$$

This greatly improves the behaviour for multiple roots with respect to the original A&B method, while not damaging much the simple roots case.

For simple roots this method has a computational efficiency near 1.70, because it is a derivative of A&B (the original authors expose $\xi \geq 1$ happens rarely for simple roots; this implies that normally $\gamma = 1 - \xi$).

5.3. A method for multiple roots

A specific formula for multiple roots of multiplicity m, in the form of (10) can be developed. Since the secant method applied to the modified function:

$$h(x) = \operatorname{sign}(f(x)) |f(x)|^{1/m}, \tag{43}$$

has superlinear convergence rate for multiple roots [30], this can be used advantageously. By using (2), in transformed coordinates (see Theorem 1) we have:

$$z_p = z_c - \frac{1 - z_c}{\operatorname{sign}(g_1)|g_1|^{1/m} - \operatorname{sign}(g_c)|g_c|^{1/m}} \operatorname{sign}(g_c)|g_c|^{1/m}, \tag{44}$$

z_p being the new tentative abscissa. g_0, g_1, g_c and z_c are given, respectively, by (5), (6), (7) and (8).

On the other hand, **(v)** in Algorithm 1 with expression (2), in transformed coordinates, for the next step, gives us:

$$z_{p1} = z_c - \frac{1 - z_c}{\gamma g_1 - g_c} g_c, \tag{45}$$

which has to provide the same value z_p as (44). By solving $z_p = z_{p1}$ we obtain:

$$\gamma = \zeta^{1-1/m}. \tag{46}$$

By Theorem 5, near the root, $0 < \zeta < 1$, this expression gives $0 < \gamma < 1$ for $m > 1$. However, far away the root it could produce a value larger than one. In any case, for safety, it is best to use:

$$\gamma = \min(1, \zeta^{1-1/m}). \tag{47}$$

This method has a computational efficiency between 1 and 1.681, the first one for the secant step and the last for the modified step. If both steps alternate the efficiency would be 1.272, in any case superlinear.

6. Comparison of several SFRF with other methods

We have to remember that some methods need several calls to the function at each step and some more at initialization. For coherence, we compare the total number of function calls. The computational efficiencies cited by the authors of the methods are obtained under iteration conditions close to the root. In addition, each function will have different multiplicative convergence rate coefficient C (in (12)), depending on function derivatives values at the root. In practice, each function, with given values of initial a and b, has enough particularity to produce clear differences in the actual rates of convergence with respect to the theoretical ones. Nevertheless, we do not intend to make a deep comparison.

As a sample we present the results for sixty functions, shown in Table 1. The majority of them are taken from the bibliography and they have been used before in many root finder tests. The last ten have multiplicity $2 \leq m \leq 6$, as a sample for testing $m \neq 1$ cases.

All calculations have been done using extended precision (18 decimal digits, `longdouble` in `numpy`) with `python-3.6.9`, `numpy-1.17.4` and `scipy-1.3.3` (for some routines) in an i5-7400 computer at a frequency of 4 GHz, with 8 GB of RAM, and Linux Ubuntu 20.04 LTS as operating system.

The use of extended precision gives us the possibility of taking $xtol = 10^{-15}$, a value that makes e_n clearly small enough to be applicable the convergence conditions as iterations progress, but a thousand times larger than the accuracy of double precision. In the cases labelled as Bis, Ill, GIll01, Peg, A&B, F4, IRF, ABI01 and SFRFm in Table 2, a modified radius control tolerance is used, like in `brentq` method, following [5, pp. 51].

Nevertheless, starting from arbitrarily wide bracketing intervals delays the initial approach to the region close enough to the root for fulfilling the convergence properties in each method. This leads to the fact that using a less demanding tolerance, *xtol*, does not significantly diminish the number of function calls.

Our aim is not a deep comparison among different methods because our three new methods are a sample for applying the common framework. We will briefly comment on the results of the new methods, shown in Table 2.

We start with the simple roots case. For simple roots GIll01 is, obviously, not competitive compared to `brentq` and `toms748`: it is not designed for it. ABI01, based on A&B, which has a computational efficiency near 1.70, is a good contender with `brentq` and `toms748`, with similar results. SFRFm in Table 2 for simple roots uses a fictitious multiplicity $m = 1.2$ in expression (47). The results are worse than for ABI01, `brentq` and `toms748`, but not by much.

For simple roots, a global comparison (for the functions presented) can be done with the average of the total number of function calls for simple root cases. They are 12.6 for SFRFm, 11.7 for `toms748`, 11.3 for ABI01 and 10.7 for `brentq`. As we see, these new methods behave similarly to those widely used in numerical packages.

For multiple roots GIll01 and ABI01 perform approximately like pure bisection, with a clear performance deterioration for increasing multiplicity. GIll01 and ABI01, for $2 \leq m \leq 6$, are better than `brentq` by a factor between about 1.5 and 2.5 and between 1.3 and 2.0 than `toms748`. When knowing the multiplicity (either integer or not) our SFRFm method is clearly of interest, because it outperforms bisection in all cases.

Although the use of the average is customary in comparing among numerical methods, in [19] a better methodology is shown, by defining a benchmark in terms of a set $\mathcal{P}$ of problems, a set $\mathcal{S}$ of solvers, and a convergence test $\mathcal{T}$.

We use a performance measure $nfun_{p,s} > 0$ (in our case the number of function evaluations required to satisfy the desired convergence) obtained for each $p \in \mathcal{P}$, $s \in \mathcal{S}$. For any tuple (p, s) the performance ratio is defined by:

$$r_{p,s} = \frac{nfun_{p,s}}{\min\{nfun_{p,s} \mid s \in \mathcal{S}\}}. \tag{48}$$

Table 1
Function definitions used in this article. The multiplicity of the function is m and i is a number for function labelling. For functions *1* and *37* we change the bracketing limits to displace the root from the centre of the interval.

i	$f(x)$	a	b	m	Reference
1	$x^3 - 1$	−0.40	1.50	1	[1], #1, new limits
2	$11x^{11} - 1$	0.10	1.00	1	[1], #3
3	$\log x$	0.50	5.00	1	[33], #1
4	$\arctan x$	−1.00	5.00	1	[33], #2
5	$x - \exp(\sin x) + 1$	1.00	4.00	1	[33], #3
6	$x\exp(-x) - 0.1$	0.00	1.00	1	[33], #4
7	$x^{1/3} - 1$	0.00	5.00	1	[33], #5
8	$x^2 - \sin^2 x -$	−1.00	2.00	1	[29], #15
9	$3x^2 - 11.12x + 9.1389$	−30.00	2.00	1	[23], #2
10	$x^6 - 36x^5 + 450x^4 - 2400x^3 + 5400x^2 - 43200x + 720$	10.00	22.00	1	[23], #4
11	$x^2(x^2/3 + \sqrt{2}\sin x) - \sqrt{3}/18$	0.10	1.00	1	[1], #2
12	$x^3 + 1$	−1.80	0.00	1	[1], #4
13	$x^3 - 2x - 5$	0.00	3.00	1	[22], Group A 18
14	$2x\exp(-5) + 1 - 2\exp(-5x)$	0.00	1.00	1	[3], Ex. 2 (n = 5)
15	$2x\exp(-10) + 1 - 2\exp(-10x)$	0.00	1.00	1	[3], Ex. 2 (n=10)
16	$2x\exp(-20) + 1 - 2\exp(-20x)$	0.00	1.00	1	[3], Ex. 2 (n=20)
17	$(1 + (1 - 5^2))x^2 - (1 - 5x)^2$	0.00	1.00	1	[3], Ex. 3 (n=5)
18	$(1 + (1 - 10^2))x^2 - (1 - 10x)^2$	0.00	1.00	1	[3], Ex. 3 (n=10)
19	$(1 + (1 - 20^2))x^2 - (1 - 20x)^2$	0.00	1.00	1	[3], Ex. 3 (n=20)
20	$x^2 - (1 - x)^5$	0.00	1.00	1	[3], Ex. 4 (n=5)
21	$x^2 - (1 - x)^{10}$	0.00	1.00	1	[3], Ex. 4 (n = 10)
22	$x^2 - (1 - x)^{20}$	0.00	1.00	1	[3], Ex. 4 (n=20)
23	$(1 + (1 - 5)^4)x - (1 - 5x)^4$	0.00	1.00	1	[3], Ex. 5 (n=5)
24	$(1 + (1 - 10)^4)x - (1 - 10x)^4$	0.00	1.00	1	[3], Ex. 5 (n=10)
25	$(1 + (1 - 20)^4)x - (1 - 20x)^4$	0.00	1.00	1	[3], Ex. 5 (n=20)
26	$(x - 1)\exp(-5x) + x^5$	0.00	1.00	1	[3], Ex. 6 (n=5)
27	$(x - 1)\exp(-10x) + x^{10}$	0.00	1.00	1	[3], Ex. 6 (n=10)
28	$(x - 1)\exp(-20x) + x^{20}$	0.00	1.00	1	[3], Ex. 6 (n=20)
29	$x^2 + \sin(x/5) - 1/4$	0.00	1.00	1	[1], #10 (n=5)
30	$x^2 + \sin(x/10) - 1/4$	0.00	1.00	1	[1], #10 (n=10)
31	$x^2 + \sin(x/20) - 1/4$	0.00	1.00	1	[1], #10 (n=20)
32	$\sin x - x^3 - 1$	−2.00	−1.00	1	[32], p. 118, ej. 3a
33	$x - \log x - 3$	2.00	6.00	1	[32], p. 118, ej. 3b
34	$(x - 1)(x - 2)(x - 3)(x - 4)(x - 5)(x - 6)$	3.10	4.50	1	[22], Group A 1d
35	$\sin x$	1.00	6.00	1	[22], Group A 2a
36	$(x^2 + 1)\sin x - \exp(\sqrt{\lvert x\rvert})(x - 1)(x^2 - 5)$	0.00	1.00	1	[22], Group A 3
37	$\frac{x+1}{x^2+2}$	−2.30	0.50	1	[22], Group A 4, new limits
38	$x^2 - 1$	−1.50	0.00	1	[22], Group A 5a
39	$x^9 + x$	−0.75	0.50	1	[22], Group B-II 1c
40	$x^{19} + x$	−0.75	0.50	1	[22], Group B-II 1d
41	$x^5 + x + 0.0001$	−0.75	0.50	1	[22], Group B-II 3b
42	$4\cos(x) - \exp(x)$	−1.00	3.00	1	[13], #1 range 2
43	$\sum_{i=1}^{10}\{\exp(xt_i) - \exp(5t_i)\}$, where $t_i = 0.1i$	4.00	6.50	1	[13], #2 range 1
44	$10^{10}x^{1/x} - 1$	0.08	0.50	1	[13], #6 range 3
45	$\sqrt{x} - 3 - 1/x$	5.00	30.00	1	[9], #7
46	$(15x - 1)/(14x)$	0.01	1.00	1	[3], Ex. 7 (n = 15)
47	$(20x - 1)/(19x)$	0.01	1.00	1	[3], Ex. 7 (n = 20)
48	$x^{1/5} - 5^{1/5}$	1.00	100.00	1	[2], #12 (n = 5)
49	$x^{1/10} - 10^{1/10}$	1.00	100.00	1	[2], #12 (n = 10)
50	$x^{1/20} - 20^{1/20}$	1.00	100.00	1	[2], #12 (n = 20)
51	$(\log x)^2\,\mathrm{sign}(x - 1)$	0.50	5.00	2	This work
52	$(x^2\exp(x) - \sin(x) + x)\,\mathrm{sign}(x)$	−0.20	5.00	2	[15], f_3 modified

(*continued on next page*)

Table 1 (*continued*).

i	$f(x)$	a	b	m	Reference
53	x^3	−0.50	1/3	3	[29], #17
54	$\left[\arctan\frac{\sqrt{5}}{2} - \arctan\sqrt{x^2-1} + \sqrt{6}\left(\arctan\sqrt{\frac{x^2-1}{6}} - \arctan\left(\frac{1}{2}\sqrt{\frac{5}{6}}\right)\right) - \frac{11}{63}\right]^3$	1.50	2.00	3	[27], ex. 3, modified
55	$x^2 \sin^2 x \operatorname{sign}(x)$	−2.00	1.00	4	This work
56	$\operatorname{sign}(x-2)(x-2)^4/((x-1)^2+1)$	1.50	2.40	4	[24], ex. 1, modified
57	x^5	−0.50	1/3	5	[29], #18
58	$(\exp(-x) - 1 + x/5)^5$	4.00	5.20	5	[4], ex. 4, modified
59	$x^3 \sin^3 x \operatorname{sign}(x)$	−1.00	0.50	6	This work
60	$\operatorname{sign}(x-2)(x-2)^6/((x-1)^2+1)$	1.90	2.20	6	[24], ex. 1, modified

The performance profile of a solver $s \in \mathcal{S}$ is the probability distribution for the ratio $r_{p,s}$. It is defined as the fraction of problems where the performance ratio is at most ω, that is:

$$\rho_s(\omega) = \frac{\text{size}\{p \in \mathcal{P} \mid r_{p,s} \leq \omega\}}{\text{size}\{p \in \mathcal{P}\}}. \tag{49}$$

An important property of performance profiles is that they are insensitive to the results on a small number of problems [10].

In Fig. 2 the results for some root finders for the whole set of functions included in Table 1, calculated from data in the last five columns of Table 2, are shown. Obviously other sets of functions, initial bracketing intervals and specific values of *xtol* would give somewhat different results.

The value of $\rho_s(1)$ is the probability that the solver will win over the rest of the solvers. Thus, if we are interested only in the number of wins, `brentq` method performs equal to or better than the others 50% of the time, and is therefore preferable. However, ABI01 is equal to or better than the others 36% of the time, usually better than and preferable to `toms748` (with $\rho_s(1) = 0.22$). For simple roots the choice is in order: `brentq`, ABI01, `toms748`.

However, both `brentq` and `toms748` are not designed for multiple roots (they lose the superlinear order and they switch to conservative linear convergence strategies). The effect of multiple roots cases in the comparison appears around $\omega = 1.35$. From this values on, `brentq` and `toms748` perform worse than SFRFm and ABI01, and our new routines would be the normal choice.

Therefore, if we do not know if we are searching a simple root, by comparing both the average number of functions needed and the performance profiles, our ABI01 routine is preferable even to `brentq`. For simple roots, obviously, `brentq` is the normal choice.

7. Conclusions

We have developed a common framework for the SFRF family of methods obtained by modification of Regula Falsi, which allows a comprehensive and rigorous comparison of them all.

We have proven that SFRF methods depend only on two dimensionless parameters, and we have shown their properties as iterations approach the root.

We have demonstrated the global convergence properties of all SFRF methods (not done before): the approximation x_n to the root α, and the radius of the bracketing interval $[a_n, b_n]$ to zero, both for simple and multiple roots. We have shown that the only mechanism needed to make the radius of the interval go to zero in SFRF methods is to use a small value for γ, the scaling factor, to force a sign change in the function for a further approximation.

The common framework allows us to develop other SFRF methods: just change the function, $F(\xi, \zeta)$, that defines the method. This simple methodology can be used to improve all known SFRF methods.

Besides, several SFRF methods can be directly combined in the same algorithm taking the advantages of each component of the combination.

Furthermore, the values given by the formulas for γ prepared to solve simple root cases tend to unity, but they tend to another value less than unity in the multiple root case. This allows a mechanism to determine the multiplicity of the root.

As an example, three new methods have been presented. They have been compared with other previously known, some devised as Regula Falsi modifications, some included in current "first class" numerical libraries. We use both the usual *nfun* average and the performance profiles for comparison.

Table 2
Total number of function calls by different methods. Bis: Bisection; Ill: Illinois; Peg: Pegasus; A&B: Anderson and Björck; F4: method 4 from [13], limited to a maximum of $\gamma = 1$, and $\gamma = 0.5$ for $\xi \geq 1$; IRF: from [21]; AC: from [33]; ST: STFA from [31]; EX: EXRF from [6]; NL: RFNL from [23]; RP and RBP: from [29]; `brentq`: function `scipy.optimize.brentq`; `toms748`: function `scipy.optimize.toms748`; GIll01: this article eq. (39); ABI01: this article eq. (42); SFRFm: this article eq. (47) —for simple root it uses a fictitious multiplicity $m = 1.2$—. For all analysed cases $xtol = 10^{-15}$, except $xtol = 10^{-12}$ for functions *9* and *10* (with the aim to compare with RFNL method in [23]). $ftol = 10^{-100}$ for all cases, except AC, STFA, EXRF, RFNL, RB and RBP (see the original articles on this subject). Data of columns AC, ST, EX, NL, RB and RBP are taken from the specific references. The rest are calculated by us. — indicates no data, *** indicates *nfun* > 999.

i	Bis	Ill	Peg	A&B	F4	IRF	AC	ST	EX	NL	RP	RBP	`brentq`	`t748`	GIll01	ABI01	SFRFm
1	53	13	11	11	11	14	—	*9*	—	—	*15*	*13*	11	12	16	11	14
2	52	17	16	25	14	20	—	*23*	*18*	—	*17*	*17*	12	17	20	16	18
3	54	11	10	10	10	15	*18*	—	*16*	—	*15*	*15*	10	10	14	10	12
4	55	10	10	10	9	15	*22*	—	—	—	—	—	9	11	10	10	9
5	54	12	12	13	12	14	*26*	—	*24*	—	*17*	*13*	14	13	16	13	13
6	52	12	9	10	10	29	*22*	—	*14*	—	—	—	9	9	13	10	11
7	55	12	11	10	12	113	*18*	—	—	—	—	—	10	7	14	10	13
8	54	14	12	13	12	15	—	—	—	—	*17*	*13*	11	13	16	13	14
9	47	16	14	16	16	14	—	—	—	*30*	—	—	20	15	17	13	15
10	46	17	15	15	19	104	—	—	—	*22*	—	—	13	16	20	17	15
11	52	13	14	14	14	***	—	—	—	—	—	—	13	14	17	12	16
12	53	13	11	13	9	14	—	—	—	—	—	—	10	11	16	12	13
13	54	14	12	12	12	14	—	—	—	—	—	—	11	15	16	12	14
14	52	13	11	11	11	14	—	*10*	—	—	—	—	11	14	15	11	12
15	52	13	11	13	12	16	—	—	—	—	—	—	12	11	15	11	13
16	52	12	11	14	13	15	—	—	—	—	—	—	13	13	15	11	14
17	52	11	10	10	10	13	—	—	—	—	—	—	10	11	13	10	11
18	52	10	10	9	12	14	—	—	—	—	—	—	10	11	13	12	12
19	52	10	13	12	11	12	—	—	—	—	—	—	9	9	13	12	12
20	52	12	13	12	12	14	—	—	—	—	—	—	9	13	15	11	11
21	52	11	12	11	13	18	—	*8*	—	—	—	—	11	12	16	11	14
22	52	13	15	14	13	***	—	—	—	—	—	—	13	14	16	14	15
23	52	10	9	9	8	10	—	—	—	—	—	—	8	11	10	9	10
24	52	9	8	8	7	10	—	—	—	—	—	—	7	11	9	8	8
25	52	9	8	8	8	8	—	—	—	—	—	—	7	7	9	8	8
26	52	11	11	9	10	***	—	—	—	—	—	—	9	12	13	9	12
27	52	15	18	10	12	***	—	—	—	—	—	—	10	13	15	12	15
28	52	23	25	14	13	***	—	—	—	—	—	—	13	15	20	15	22
29	52	12	10	10	10	14	—	—	—	—	—	—	11	11	14	10	12
30	52	12	10	10	10	15	—	—	—	—	—	—	11	11	14	10	12
31	52	13	10	10	10	15	—	—	—	—	—	—	11	13	14	10	12
32	52	11	11	11	11	14	—	—	—	—	—	—	9	10	15	11	13
33	54	9	8	8	8	21	—	—	—	—	—	—	8	9	10	8	9
34	53	12	10	11	10	13	—	—	—	—	—	—	10	12	11	11	12
35	55	12	12	13	12	***	—	—	—	—	—	—	10	15	15	10	11
36	52	12	11	11	10	12	—	—	—	—	—	—	9	11	11	10	11
37	54	12	10	11	11	56	—	—	—	—	—	—	11	10	16	11	13
38	53	11	9	10	9	13	—	—	—	—	—	—	10	8	14	10	12
39	53	7	9	8	9	12	—	—	—	—	—	—	8	8	8	8	7
40	53	6	7	7	8	10	—	—	—	—	—	—	7	7	6	7	5
41	53	7	9	8	8	12	—	—	—	—	—	—	9	9	9	8	7
42	54	14	13	13	13	15	—	—	—	—	—	—	12	10	16	13	15
43	54	12	10	10	10	27	—	—	—	—	—	—	10	11	15	10	13
44	51	39	38	557	26	116	—	—	—	—	—	—	20	18	29	26	24
45	57	12	10	9	10	51	—	—	—	—	—	—	8	11	13	9	11
46	52	16	15	5	14	23	—	—	—	—	—	—	14	19	19	9	21
47	52	16	13	5	14	23	—	—	—	—	—	—	14	18	19	8	19

(*continued on next page*)

Table 2 (*continued*).

i	Bis	Ill	Peg	A&B	F4	IRF	AC	ST	EX	NL	RP	RBP	`brentq`	`t748`	GIll01	ABI01	SFRFm
48	59	13	11	10	12	97	—	—	—	—	—	—	11	10	15	10	14
49	59	13	11	11	12	149	—	—	—	—	—	—	11	11	15	11	15
50	59	12	11	10	11	205	—	—	—	—	—	—	11	11	15	10	15
51	54	65	116	93	***	***	—	—	—	—	—	—	75	96	42	45	17
52	55	77	126	93	***	***	—	—	—	—	—	—	142	104	40	49	19
53	52	220	357	270	270	***	—	—	—	—	***	*91*	129	107	54	52	8
54	51	123	194	148	145	***	—	—	—	—	—	—	122	104	43	57	18
55	54	250	391	289	289	***	—	—	—	—	—	—	137	126	80	68	16
56	52	186	288	216	***	***	—	—	—	—	—	—	124	112	58	70	22
57	52	262	400	292	293	***	—	—	—	—	***	*101*	129	127	81	91	13
58	53	249	374	274	256	***	—	—	—	—	—	—	126	114	72	87	20
59	53	276	416	301	301	***	—	—	—	—	—	—	154	141	85	95	19
60	51	265	399	288	289	***	—	—	—	—	—	—	146	129	82	91	20

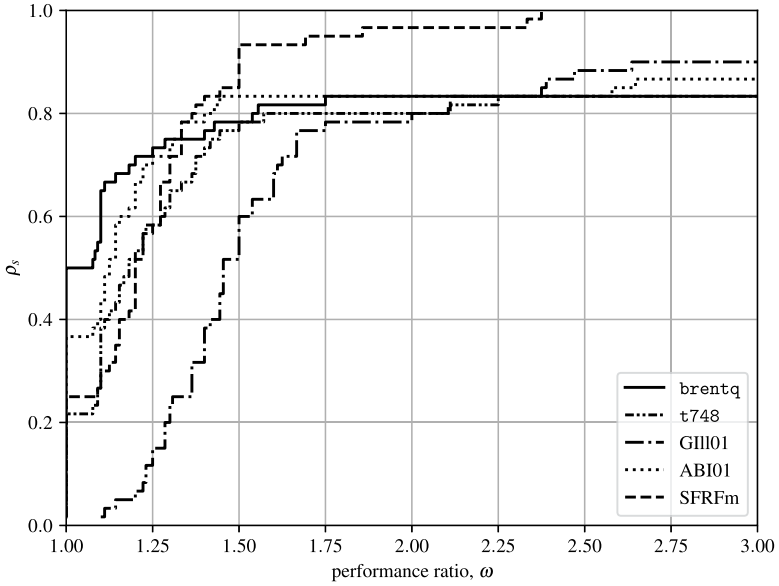

Fig. 2. Cumulative distribution for the performance profile $\rho_s(\omega)$ for `brentq`, `toms748`, Gill01, ABI01 and SFRFm methods.

One of the new methods (ABI01) improves the one of Anderson and Björck. For simple roots, it compares well with the methods previously developed. However, for multiple roots of unknown multiplicity ABI01 outperforms the others, even some which belong to numerical libraries widely used by the scientific community. Since normally we do not know the multiplicity of a root, ABI01 could be used, with confidence, as a commitment method for unknown multiplicity.

For multiple roots another method (SFRFm) has been presented. Until now, for this case no specific method of SFRF kind has been proposed. When multiplicity is known this new method beats even bisection, as it is superlinear. For example, this can be used for some polynomials.

In any case, new methods, adequate for single and multiple roots, could be obtained in the future by using the developed framework.

CRediT authorship contribution statement

Julio M. Fernández-Díaz: Conceptualization, Formal analysis, Methodology, Writing original, Computer programming. **César O. Menéndez-Pérez:** Conceptualization, Formal analysis, Methodology, Writing original.

Acknowledgements

The authors would like to thank the anonymous reviewers for their help to improve the final version of this manuscript.

Appendix. Pseudocodes

```
function GIll01 (f, a, b, xtol, ftol, nfunmax)
   EPSmachine = 2.22e-16
   fa = f(a), fb = f(b), nfun = 2
   if fa*fb >= 0 then
      return a, nfun, "No bracketing"
   endif
   c = b, fc = fb
   while nfun < nfunmax do
      xtol1 = 2*EPSmachine*abs(c)+xtol
      if |b-a| < xtol1 then
         return c, nfun, "xtol reached"
      endif
      c = c-(b-a)/(1.0-fa/fb), fc = f(c), nfun = nfun+1
      if |fc| < ftol then
         return c, nfun, "ftol reached"
      endif
      if fc*fb < 0 then
         a = b, fa = fb
         gamma = 0.1, fa = fa*gamma
      endif
      b = c, fb = fc
   enddo
   return c, nfun, "maximum number of functions reached"
endfunction
```

```
function ABI01 (f, a, b, xtol, ftol, nfunmax)
   EPSmachine = 2.22e-16
   mulkind = "single root"
   fa = f(a), fb = f(b), nfun = 2
   if fa*fb >= 0 then
      return a, nfun, "No bracketing", "unknown multiplicity"
   endif
   nbis0 = 1+floor(log2(|a-b|/xtol)/3)
   c = b, fc = fb
   while nfun < nfunmax do
      xtol1 = 2*EPSmachine*abs(c)+xtol
      if |b-a| < xtol1 then
         return c, nfun, "xtol reached", mulkind
      endif
      c = c-(b-a)/(1.0-fa/fb), fc = f(c), nfun = nfun+1
      if |fc| < ftol then
         return c, nfun, "ftol reached", mulkind
      endif
      if fc*fb < 0 then
         a = b, fa = fb
      elseif nfun >= nbis0 then
         gamma = 0.1, fa = fa*gamma
         mulkind = "(probable) multiple root"
```

```
      else
         xi = fc/fb
         gamma = max(1-xi, 0.1), fa = fa*gamma
      endif
      b = c, fb = fc
   enddo
   return c, nfun, "maximum number of functions reached", mulkind
endfunction
```

```
function SFRFm (f, a, b, xtol, ftol, nfunmax, mul)
   EPSmachine = 2.22e-16
   if mul == 1 then
      mul = 1.2
  endif
   fa = f(a), fb = f(b), nfun = 2
   if fa*fb >= 0 then
      return a, nfun, "No bracketing"
   endif
   c = b, fc = fb
   while nfun < nfunmax do
      xtol1 = 2*EPSmachine*abs(c)+xtol
      if |b-a| < xtol1 then
         return c, nfun, "xtol reached"
      endif
      c = c-(b-a)/(1.0-fa/fb), fc = f(c), nfun = nfun+1
      if |fc| < ftol then
         return c, nfun, "ftol reached"
      endif
      if fc*fb < 0 then
         a = b, fa = fb
      else
         zz = -fc/fa
         gamma = min(1, zz^(1-1/mul)), fa = fa*gamma
      endif
      b = c, fb = fc
   enddo
   return c, nfun, "maximum number of functions reached"
endfunction
```

References

[1] G.E. Alefeld, F.A. Potra, Some efficient methods for enclosing simple zeros of nonlinear equations, BIT 32 (1992) 334–344, http://dx.doi.org/10.1007/BF01994885.

[2] G.E. Alefeld, F.A. Potra, Y. Shi, Algorithm 748: Enclosing zeros of continuous functions, ACM Trans. Math. Software 221 (1995) http://dx.doi.org/10.1145/210089.210111.

[3] N. Anderson, Å. Björck, A new high order method of regula falsi type for computing a root of an equation, BIT 13 (1973) 253–264, http://dx.doi.org/10.1007/BF01951936.

[4] R. Behl, A. Cordero, J.R. Torregrosa, A new higher-order optimal derivative free scheme for multiple roots, J. Comput. Appl. Math. 404 (3) (2022) 113773, http://dx.doi.org/10.1016/j.cam.2021.113773.

[5] R.P. Brent, Algorithm for Minimization Without Derivatives, Prentice-Hall, Inc., Englewood Cliffs, N.J, 1973.

[6] J. Chen, W. Li, An improved exponential regula falsi methods with quadratic convergence of both diameter and point for solving nonlinear equations, Appl. Numer. Math. 57 (2007) 80–88, http://dx.doi.org/10.1016/j.apnum.2006.01.001.

[7] J. Chen, J. Shen, On third-order convergent regula falsi method, Appl. Math. Comput. 188 (2007) 1592–1596, http://dx.doi.org/10.1016/j.amc.2006.11.030.

[8] X.-D. Chen, J. Shi, W. Ma, A fast and robust method for computing real roots of nonlinear equations, Appl. Math. Lett. 68 (2017) 27–32, http://dx.doi.org/10.1016/j.aml.2016.12.013.

[9] C. Chun, M.Y. Lee, A new optimal eighth-order family of iterative methods for the solution of nonlinear equations, Appl. Math. Comput. 223 (2013) 506–519, http://dx.doi.org/10.1016/j.amc.2013.08.033.
[10] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213, http://dx.doi.org/10.1007/s101070100263.
[11] M. Dowell, P. Jarratt, A modified regula falsi method for computing the root of an equation, BIT 11 (1971) 168–174, http://dx.doi.org/10.1007/BF01934364.
[12] M. Dowell, P. Jarratt, The Pegasus method for computing the root of an equation, BIT 12 (1972) 503–508, http://dx.doi.org/10.1007/BF01932959.
[13] J.A. Ford, Improved Algorithms of Illinois-Type for the Numerical Solution of Nonlinear Equations, Technical Report CSM-257, University of Essex, 1995.
[14] S. Galdino, A family of regula falsi root-finding methods, in: Proceedings of 2011 World Congress on Engineering and Technolgy (Vol. 1), Pages 514–517, Shanghai, China, 2011.
[15] J. Hueso, E. Martínez, C. Teruel, Determination of multiple roots of nonlinear equations and applications, J. Math. Chem. 53 (2015) 880–892, http://dx.doi.org/10.1007/s10910-014-0460-8.
[16] R.F. King, Methods without secant steps for finding a bracketed root, Computing 17 (1976) 49–57, http://dx.doi.org/10.1007/BF02252259.
[17] V. Kodnyanko, Improved bracketing parabolic method for numerical solution of nonlinear equations, Appl. Math. Comput. 400 (2021) 125995, http://dx.doi.org/10.1016/j.amc.2021.125995.
[18] L. Kogan, L. Sapir, Amir Sapir, Ariel Sapir, The Fibonacci family of iterative processes for solving nonlinear equations, Appl. Numer. Math. 68 (2016) 148–158, http://dx.doi.org/10.1016/j.apnum.2016.08.012.
[19] J.J. Moré, S.M. Wild, Benchmarking derivative-free optimization algorithms, SIAM J. Optim. 20 (1) (2009) 172–191, http://dx.doi.org/10.1137/080724083.
[20] D.E. Muller, A method for solving algebraic equations using an automatic computer, MTAC 10 (1956) 208–215, http://dx.doi.org/10.2307/2001916.
[21] J. Naghipoor, S.A. Ahmadian, A.R. Soheili, An improved regula falsi method for finding simple zeros of nonlinear equations, Appl. Math. Sci. 2 (8) (2008) 381–386, http://dx.doi.org/10.12988/ams.
[22] D. Nerinckx, A. Haegemans, A comparison of non-linear equation solvers, J. Comput. Appl. Math. 2 (2) (1976) 145–148, http://dx.doi.org/10.1016/0771-050X(76)90017-6.
[23] P.K. Parida, D.K. Gupta, An improved regula-falsi method for enclosing simple zeros of nonlinear equations, Appl. Math. Comput. 177 (2006) 769–776, http://dx.doi.org/10.1016/j.amc.2005.11.034.
[24] P.K. Parida, D.K. Gupta, An improved method for finding multiple roots and it's multiplicity of nonlinear equations in $\mathbb{R}$, Appl. Math. Comput. 202 (2008) 498–503, http://dx.doi.org/10.1016/j.amc.2008.02.030.
[25] M.S. Petković, On a general class of multipoint root-finding methods of high computational efficiency, SIAM J. Numer. Anal. 47 (6) (2010) 4402–4424, http://dx.doi.org/10.1137/090758763.
[26] A. Ralston, P. Rabinowitz, A First Course in Numerical Analysis, second ed., Dover Pub. Inc., New York, 2001.
[27] J.R. Sharma, S. Kumar, I.K. Argyros, Development of optimal eighth order derivative-free methods for multiple roots of nonlinear equations, Symmetry 11 (6) (2019) http://dx.doi.org/10.3390/sym11060766.
[28] J.N. Snyder, Library Routine H1-71, University of Illinois Digital Computer Laboratory, 1953.
[29] A. Suhadolnik, Combined bracketing methods for solving nonlinear equations, Appl. Math. Lett. 25 (2012) 1755–1760, http://dx.doi.org/10.1016/j.aml.2012.02.006.
[30] J.F. Traub, Iterative Methods for the Solution of Equations, Prentice-Hall, Inc., Englewood Cliffs, N.J, 1964.
[31] X. Wu, Z. Shen, J. Xia, An improved regula falsi method with quadratic convergence of both diameter and point for enclosing simple zeros of nonlinear equations, Appl. Math. Comput. 144 (2003) 381–388, http://dx.doi.org/10.1016/S0096-3003(02)00414-9.
[32] D.M. Young, R.T. Gregory, A Survey of Numerical Mathematics, Dover Pub. Inc., New York, 1988.
[33] Y. Zhu, X. Wu, A free-derivative iteration method of order three having convergence of both point and interval for nonlinear equations, Appl. Math. Comput. 137 (2003) 49–55, http://dx.doi.org/10.1016/S0096-3003(02)00029-2.