Journal of Mathematical Modeling
Vol. 11, No. 4, 2023, pp. 631–647. Research Article

JMM

New algorithms to estimate the real roots of differentiable functions and polynomials on a closed finite interval

Hassan Khandani*

Department of mathematics, Mahabad Branch, Islamic Azad university, Mahabad, Iran
Email(s): Hassan.Khandani@iau.ac.ir

Abstract. We propose an algorithm that estimates the real roots of differentiable functions on closed intervals. Then, we extend this algorithm to real differentiable functions that are dominated by a polynomial. For each starting point, our method converges to the nearest root to the right or left hand side of that point. Our algorithm can look for missed roots as well and theoretically it misses no root. Furthermore, we do not find the roots by randomly chosen initial guesses. The iterated sequences in our algorithms converge linearly. Therefore, the rate of convergence can be accelerated considerably to make it comparable to Newton-Raphson and other high-speed methods. We have illustrated our algorithms with some concrete examples. Finally, the pseudo-codes of the related algorithms are presented at the end of this manuscript.

Keywords: Krasnoselskii sequence, iterative method, Newton-Raphson method, root estimation, real function.
AMS Subject Classification 2010: 26A18, 49M15, 65-02.

1 Introduction and preliminaries

In this manuscript, first we present an algorithm to find the roots of a differentiable function $g:[a,b]\to\mathbb{R}$ such that $|g^{'}(x)|\leq 2$ for each $x\in[a,b]$. This algorithm, moves forward from a to b and finds one root after another until all roots in $[a,b]$ are found. Then, we equip this algorithm with a function that is called the missed-root function to report about the missed roots as well. Then, we extend this algorithm to be applied to functions g dominated by a polynomial without assuming the condition $|g^{'}|\leq 2$ to be hold on $[a,b]$ (see Algorithm 3 and 4). It is worth mentioning that to be dominated by a polynomial is not a strong condition when g is a polynomial or when $g(x)=a_n(x)x^n+\cdots+a_1(x)x+a_0(x)$, where $a_i(x)$ are bounded differentiable functions for $i=1,2,\ldots,n$ (see Example 4 and 5). Then, we show that our

*Corresponding author
Received: 30 September 2022 / Revised: 18 July 2023 / Accepted: 19 July 2023
DOI: 10.22124/JMM.2023.22967.2040

sequences converge linearly. This shows that the rate of convergence of our algorithm can be accelerated that can make it comparable to other high-speed methods (see Section 4 for more details).

Our method can be regarded as a root-finding method. Because of that, we take a brief look at some root-finding methods as follows. There are many different algorithms based on iterations such as, Dekker-Brent algorithm [4], Chandrupatlas method [7], Ridders method [23], Alefeld-Potra method [1], and more. For a complete discussion about root-finding methods and their significance in pure and applied mathematics, we refer the reader to [20], [22], [12]. Newton-Raphson method and most other similar methods guarantee their convergence at the local level by Taylor's theorem. For this reason, one way of estimating a root is to isolate each root in a small interval. Then, Newton-Raphson or some other methods are applied to a starting point in these isolated small intervals to find the related root. In most root-finding methods for polynomials, the isolation of the roots is an important step. Sturm's theorem, Descartes' rule of signs, Vincent theorem, Vincent's auxiliary theorem, and some other generalizations of these results estimate the number of roots of a polynomial in an interval [24], [26]. Some of these results or a combination of them can provide a real-root isolation algorithm [9].

In dealing with other functions rather than polynomials, there exist no similar results for the isolation of the roots. In such cases, a set of starting points is chosen, and at each point of this set Newton-Raphson or some other similar methods are applied to find the roots. In applying our method, we neither use a set of starting points nor any root-isolation process to look for the roots. In Example 3, we define the relative time efficiency of a method and based on that make a comparison between our method and Newton-Raphson method. In this example, we show that the relative time efficiency for our method to find all related roots is about 98%, while the estimated relative time efficiency of Newton-Raphson method is less than 2% (see Table 2). This shows that our method for differentiable functions is more suitable than Newton-Raphson method. As we mentioned before, our method can be used to find the real roots of real polynomials. We have also presented the related algorithm to estimate the roots of polynomials or differentiable functions dominated by a polynomials.

The following results will be needed in the rest of this manuscript.

Definition 1. *Let $L > 0$, $g : [a,b] \to \mathbb{R}$ ia called an L-Lipschitz mapping if $|g(x) - g(y)| \leq L|x-y|$ for each $x, y \in [a,b]$.*

Definition 2. *(Krasnoselskii's sequence [8]) Let X be a Banach space, A be a non empty subset of X, and $g : A \to A$ be an arbitrary mapping. For each $x_0 \in A$ and $\{t_n\}_{n=0}^{\infty} \subset [0,1]$, the Mann iteration of g is defined as follows:*

$$x_{n+1} = t_n x_n + (1-t_n) g(x_n) \text{ for each } n \geq 0. \tag{1}$$

If $t_n = \frac{1}{2}$ for each $n \geq 0$, the sequence $\{x_n\}$ is called the Krasnoselskii iteration of g. If $t_n = t$ for each $n \geq 0$, where $t \in [0,1]$, then this sequence is called the Schaefer iteration of g.

We have the following fixed point result for non-self mappings on non-empty convex subsets of Banach spaces.

Theorem 1. *(Krasnoselskii [18]) Suppose A is a convex compact subset of an uniformly convex Banach space $(X, ||.||)$. For every nonexpansive mapping $g : A \to A$ (i.e., $||g(x) - g(y)|| \leq ||x-y||$ for each $x, y \in A$) the sequence of iterations defined by Equation 1 converges to a fixed point of g.*

Bailey gave a proof of Krasnoselskii's result for nonexpansive real-valued functions on a closed interval [2]. Hillam B. P. [15] extended Bailey's result to Lipschitz functions. Borwein and Borwein extended Hillam's result to Mann iterations [3].

Theorem 2. *(Hillam [15]) Let $L > 0$, and $h : [a,b] \to [a,b]$ be an L-Lipschitz mapping. Then, the sequence of iterations defined by Equation (1) converges monotonically to a fixed point of h if $0 < t \leq \frac{1}{1+L}$.*

Khandani et. al. studied the convergence of the Krasnoselskii sequence of differentiable functions. They provided the following root-finding results that are essential in our study.

Theorem 3. *[17] Let g be a continuous real-valued function on $[a,b]$ and $c \in (a,b)$ be the unique root of g in $[a,b]$. Suppose that g is differentiable on $A = (a,c) \cup (c,b)$ with $g^{'}(x) \geq -2$ for each $x \in A$. Also, suppose that $g(x) > 0$ for each $x \in [a,c)$ and $g(x) < 0$ for each $x \in (c,b]$. For each $x_0^+ \in [a,b]$ and for each $n \geq 0$ define:*

$$x_{n+1}^+ = x_n^+ + \frac{g(x_n^+)}{2}, \tag{2}$$

then the sequence $\{x_n^+\}$ converges to c.

Theorem 4. *[17] Let g be a continuous real-valued function on $[a,b]$ and $c \in (a,b)$ be the unique root of g in $[a,b]$. Suppose that g is differentiable on $A = (a,c) \cup (c,b)$ with $g^{'}(x) \leq 2$ for each $x \in A$. Also, suppose that $g(x) < 0$ for each $x \in [a,c)$ and $g(x) > 0$ for each $x \in (c,b]$. For each $x_0^- \in [a,b]$ and for each $n \geq 0$ define:*

$$x_{n+1}^- = x_n^- - \frac{g(x_n^-)}{2}, \tag{3}$$

then the sequence $\{x_n^-\}$ converges to c.

Definition 3. *[25] Let $\{x_n\}$ be a sequence of real numbers that converges to ξ and there exists $\lambda \in (0,1]$ such that*

$$\lim_{n\to\infty} \frac{|x_{n+1} - \xi|}{|x_n - \xi|} = \lambda. \tag{4}$$

If $\lambda \in (0,1)$, then $\{x_n\}$ converges to ξ linearly. If $\lambda = 1$, then $\{x_n\}$ converges to ξ sublinearly. The rate of convergence for a sublinear sequence is slower than a linearly convergent sequence.

Theorem 5 (Steffenson acceleration method, [14], [5])**.** *Let $\{x_n\}_0^\infty$ be a sequence of real numbers that converges to x linearly. For each $n \geq 0$ define $y_n = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}$. Then*

$$\lim_{n\to\infty} \frac{y_n - x}{x_n - x} = 0, \tag{5}$$

where $\Delta x_n = x_{n+1} - x_n$ and $\Delta^2 x_n = \Delta(\Delta x_n)$ and $\{y_n\}$ converges to x faster than $\{x_n\}$.

2 The algorithm for differentiable functions

To start our discussion, first, we present the following notations for the sequences introduced in Theorems 3, 4. These notations will be more useful for our study.

Notation 1. We denote the sequences $\{x_n^+\}$ and $\{x_n^-\}$ with initial point a by $\{x^+(n,g,a,)\}$ and $\{x^-(n,g,a)\}$ respectively. Since throughout our discussion g would be a fixed function, when there is no ambiguity, we drop g and denote these two sequences by $\{x^+(n,a)\}$ and $\{x^-(n,a)\}$ respectively.

In Theorems 3 and 4 the related iterations converge monotonically (see the proof of these results in [17]). We prove Proposition 1 and deduce Theorems 3 and Theorem 4 as a consequence of this result. Especially we assert that the iterations introduced in these results converge monotonically as follows.

Proposition 1. *Let $a < c < b$, $L > 0$, $g : [a,b] \to \mathbb{R}$ be a continuous mapping such that $\frac{g(x)-g(y)}{x-y} \geq -L$ for each $x, y \in [a,b]$. Also suppose that $g(x) > x$ for each $x \in [a,c)$, $g(x) < x$ for each $x \in (c,b]$, and c be the unique fixed point of g in $[a,b]$. For $x_0 \in [a,b]$ define:*

$$x_{n+1} = (1-t)x_n + tg(x_n), \forall n \geq 0 \tag{6}$$

(i) $\{x_n\}$ converges monotonically to c for each $t \in (0, \frac{1}{1+L}]$,

(ii) The sequence $\{x_n^+\}$ and $\{x_n^-\}$ in Theorem 3 and 4 respectively converges monotonically to c.

Proof. To prove (i), suppose that $x_0 \in [a,c)$. If $x_m = c$ for some non-negative integer m, then we have $x_n = c$ for each $n \geq m$ and the proof is complete. Therefore, we assume $x_n \neq c$ for each $n \geq 0$. Let $0 < t \leq \frac{1}{1+L}$ and $x \in [a,c)$. We have $\frac{g(x)-c}{x-c} = \frac{g(x)-g(c)}{x-c} \geq -L \geq 1 - \frac{1}{t}$. This follows that $\frac{g(x)-x}{x-c} \geq \frac{-1}{t}$. Rearranging this inequality, we get:

$$(1-t)(x) + tg(x) \leq c. \tag{7}$$

Assume $\{x_n\}$ be as defined by Equation (6). From (7) and argument by induction we see that $x_n < c$ for each $n \geq 0$. We show that $\{x_n\}$ is an increasing sequence. Since $g(x_0) > x_0$, we have $c > x_1 = (1-t)x_0 + tg(x_0) > x_0$. Suppose $x_0 < x_1 < \cdots < x_m < c$. Since $a \leq x_m < c$, by our assumption $g(x_m) > x_m$. It follows that $x_{m+1} = (1-t)x_m + tg(x_m) > x_m$. Therefore, by induction, $\{x_n\}$ is an increasing sequence. Now, $x_n < c$ for each $n \geq 0$ and $\{x_n\}$ is an increasing sequence in $[a,c]$. Assume $\{x_n\}$ converges to $d \in [a,c]$, where d is also a fixed point of g. Since g has a unique fixed point in $[a,b]$, we deduce that $d = c$. When $x_0 \in (c,b]$ the proof is similar, this time $\{x_n\}$ converges to c decreasingly. If $x_0 = c$, then $x_n = c$ for each $n \geq 0$ and the proof is evident.
Proof of (ii): Let g be a continuous real-valued function that satisfies in all conditions of Theorem 3, and c be the unique root of g in (a,b). Define $h(x) = g(x) + x$ for each $x \in [a,b]$. We notice that c is the unique fixed point of h in (a,b), $\frac{h(x)-h(y)}{x-y} \geq -1$ for each $x, y \in [a,b]$, and h satisfies in all conditions of part (i) with $L = 1$. Therefore, $\{x(n,h,x_0)\}$ converges monotonically to c for each $x_0 \in [a,b]$ and $t \in (0,1/2]$. We simply denote this sequence by $\{x_n\}$. Set $t = 1/2$, for each $n \geq 0$ we have:

$$\begin{aligned} x_{n+1} &= (1 - \frac{1}{2})x_n + \frac{1}{2}(g(x_n) + x_n) \\ &= x_n + \frac{g(x_n)}{2}. \end{aligned}$$

Therefore, $x_n = x_n^+$ for each $n \geq 0$. This shows that $\{x_n^+\}$ converges monotonically to c for each $x_0 \in [a,b]$. For $\{x_n^-\}$, we define $h(x) = -g(x) + x$ for all $x \in [a,b]$ and the argument is similar. □

Remark 1. *Proposition 1 holds true when g is an L-Lipschitz mapping. This shows that for L-Lipschitz mappings we can always find $t > 0$ such that $\{x_n\}$ converges to a fixed point of g, no matter how large L is. Example 1 shows that the condition $t \leq \frac{1}{1+L}$ is necessary for the convergence of $\{x_n\}$ in Proposition 1. Borwein et al. relaxed the L-Lipschitz condition and extended this result to continuous mappings (Proposition 3 Borwein et al. [3]).*

We present the following result about the sequences $\{x^-(n,a)\}$ and $\{x^+(n,a)\}$. This result clearly shows that the sign of $g(a)$ is crucial to find the roots of g that are located to the right or left hand side of a.

Lemma 1. *Suppose that $g:[a,b]\to\mathbb{R}$ be a 2-Lipschitz mapping or be a differentiable function such that $|\,g'(x)\,|\leq 2$ for each $x\in\mathbb{R}$, $a\in\mathbb{R}$ with $g(a)\neq 0$. Then,*

(i) If $g(a)>0$, then $x^+(n,a)$ converges to $c\in[a,b]$ if and only if there exists a root of g in $[a,b]$. In this case, c is the nearest root of g to a in $[a,b]$.

(ii) If $g(a)<0$, then $x^-(n,a)$ converges to $c\in[a,b]$ if and only if there exists a root of g in $[a,b]$. In this case, c is the nearest root of g to a in $[a,b]$.

(iii) If $g(b)<0$, then $x^+(n,b)$ converges to $c\in[a,b]$ if and only if there exists a root of g in $[a,b]$. In this case, c is the nearest root of g to b in $[a,b]$.

(iv) If $g(b)>0$, then $x^-(n,b)$converges to $c\in[a,b]$ if and only if there exists a root of g in $[a,b]$. In this case, c is the nearest root of g to b in $[a,b]$.

If g has no root in $[a,b]$, then each of these sequences exit this interval.

Proof. Let $g(a)>0$ and suppose that $x^+(n,a)$ converges to c as $n\to\infty$. By Proposition 1 $\{x^+(n,a)\}$ is a monotone sequence. Since $x_1=a+\frac{g(a)}{2}>a=x_0$, we deduce that $\{x^+(n,a)\}$ is an increasing sequence and $c>a$. By the definition of this sequence, we have $x^+(n+1,a)=x^+(n,a)+\frac{g(x^+(n,a))}{2}$. Since g is continuous, taking limit of this equation as $n\to\infty$ follows that $g(c)=0$ and the proof is complete. To prove the converse part of (i), let $g(a)>0$ and suppose that $c>a$ is a root of g in $[a,b]$. Without loss of generality suppose that c is the nearest root of g in $[a,b]$. By Theorem 3, $x^+(n,a)$ converges to c as $n\to\infty$, which completes the proof of part (i). The proof of other parts are similar. Assume g has no root in $[a,b]$. since each of these sequences are monotone, they exit this interval for some non-negative integer m. □

2.1 Algorithm

Let $g:[a,b]\to\mathbb{R}$ be a real-valued continuous function that is differentiable on (a,b) and $|\,g'(x)\,|\leq 2$ for each $x\in(a,b)$ (or be a 2-Lipschitz mapping). Suppose g has a finite number of roots in $[a,b]$. Then, we estimate all roots of g in [a,b] as follows.

Let $\varepsilon>0$ with $a+\varepsilon<b$ and F be the empty set. We start from a and step by step find all the roots of g in $[a,b]$ as follows: If $g(a)=0$, then a is a root of g and we add it to the set F. If $g(a)>0$ and $\lim_{n\to\infty}x^+(n,a)>b$, then by Lemma 1, g has no roots in $[a,b]$ and we should stop the process. Otherwise, $c=\lim_{n\to\infty}x^+(n,a)\leq b$ is a root of g and we add it to F and start the process again by replacing a with $c+\varepsilon$ provided that $c+\varepsilon\leq b$. In case of $g(a)<0$, set $c=\lim_{n\to\infty}x^-(n,a)$. If $c\leq b$, we add c to F and start the process again with $a=c+\varepsilon$ provided that $c+\varepsilon\leq b$, otherwise we end the process.

We calculate the $\lim_{n\to\infty}x^+(n,g,a)$ or $\lim_{n\to\infty}x^-(n,g,a)$ at each point $a\in\mathbb{R}$ via a while loop as follows:

Calculating the limit of the sequences $x^+(n,g,a)$ and $x^-(n,g,a)$. First, we take a look at the function y^+ that is defined in Line 2 in Algorithm 1. If $g(a)>0$, then $y^+(a)=a+\frac{g(a)}{2}=x^+(1,a)$. Therefore, by definition of $x^+(n,a)$, replacing a with $y^+(a)$ repeatedly while $g(a)>0$ will calculate $\lim_{n\to\infty}x^+(n,a)$. When $g(a)<0$, replacing a with $y^+(a)$, $\lim_{n\to\infty}x^-(n,a)$ is calculated. The *f_limit* function has been defined to peform this task (see line 8 of Algorithm 1). This function runs a while loop with condition

$|g(a)| > \delta$. When we are out of this while loop, we have $|g(a)| \leq \delta$, which means that a is a root of g with the precision δ. Since this function, for a given point a calculates the nearest root to a inside $[a,b]$, the name $forward-limit$ has been chosen for it that is briefly denoted by f_limit. We stop this while loop when $y^+(z) = z$, otherwise we would have an infinite while loop (see Line 11 of Algorithm 1).
We have added the condition $z \leq b$ inside this while loop. This condition does not allow the limit to grow greater than b inside this while loop. The backward-limit function is defined similarly, which at any point returns the nearest root at the left side of that point or returns a statement. We will denote the backward-limit function by b_limit. We use this function to know about the missing roots.

Making a report about the missing roots. We notice that when a root c is found we replace c with $c+\varepsilon$ to start the process again in the next step. So, all the roots in the interval $(c, c+\varepsilon)$ will be missed. We consider that if the interval $(c, c+\varepsilon)$ contains no roots of g, then by Lemma 1 $b\_limit(a+\varepsilon) = c$. Therefore, $b\_limit(a+\varepsilon) \neq c$ shows that some roots has been missed. Based on this fact, we define the missed-root function (Line 2 of Algorithm 2). Now, endowing our first algorithm with a missed-root function, we present our second algorithm (see Algorithms 2). The second algorithm will give us information about the found and missed roots of g for any given $\varepsilon > 0$.
To find the lost roots, we can make ε smaller and smaller until no lost-root is reported. This is possible because by assumption g has finite number of roots in $[a,b]$.

3 The algorithm for differentiable functions dominated by some polynomials

Before we begin our discussion, we assert that there are some issues with our method presented earlier when we are going to find the roots of polynomials with large degrees. To be more clear, let $g(x) = x^{75} - 3x^{50} + x^{25} - 2$ for each $x \in \mathbb{R}$ and the goal is to find the real roots of this polynomial in $[-2,2]$. Let $M = 75 \times 2^{74} + 150 \times 2^{49} + 25 \times 2^{24} + 1$. We replace g by g/M, then $|g^{'}(x)| \leq 2$ for each $x \in [-2,2]$. Function g satisfies all necessary conditions to apply Algorithm 1 to find its roots in this interval. But, this algorithm is extremely slow. We know that the roots are the limit of the sequences defined by Equations (2) or (3). If we simply denote each of these sequences by $\{x_n\}$, then we have

$$|x_{n+1} - x_n| \leq \frac{|g(x_n)|}{2M}.$$

The above inequality shows that when $M > 0$ is a large number x_{n+1} and x_n are very close to each other, in other words the speed becomes too slow. Setting $x_0 = 0$, we see that

$$|x_1 - x_0| \leq \frac{|g(0)|}{2M} \leq \frac{2}{2 \times 75 \times 2^{74}}.$$

Therefore, the problem is that we have used a large constant M for the entire interval $[-2,2]$. To get over these problem, the first thing that comes to our mind is to implement the divide-and-conquer method in [10]. This means that we should divide the interval $[-2,2]$ into many small-length sub-intervals and for each of these sub-intervals, we should choose a suitable constant to prevent slow convergence of the related sequences and find the roots of this polynomial in each of these sub-intervals. Finding a suitable

constant $M > 0$ for each sub-interval practically is impossible. In what follows, we do not divide the given interval manually and let the sequences do this division automatically. We modify the definitions the sequences such that each time they are called, g is divided by a suitable constant (Line 3 of Algorithm 3). We describe each step of the algorithms 3 and 4 which have been designed for this purpose. First, we present the following results that will be needed in the sequel.

Lemma 2. *Suppose $g : [a,b] \to \mathbb{R}$ be a differentiable function, $a < z < c < 0$, $g(z) \neq 0$, $p(x) = a_m x^m + \cdots + a_1 x + a_0$ be a polynomial of degree m with real coefficients such that $|g^{'}(x)| \leq p(x)$ for $x \in [a,0]$. Assume c be the nearest root of g to the right side of z. Then, one of the sequences $\{x^+(n,h,z)\}$ or $\{x^-(n,h,z)\}$ converges to c, where $h(x) = \frac{g(x)}{f(|z|)}$ for each $x \in [a,b]$, $f(x) = |a_m|x^m + \cdots + |a_1|x + |a_0| + 1$. The result holds true if we replace the condition $a < z < 0$ with $0 < z < b$.*

Proof. We notice that:

$$\frac{2|g^{'}(x)|}{f(|z|)} \leq 2 \text{ for each with } |x| \leq |z|.$$

Then, the function $h(x) = \frac{2g(x)}{f(|z|)}$ satisfies all conditions of Theorems 3 and 4 on the interval $[z,c]$. By writing the sequences defined in Theorems 3 and 4 for the function h, we see that: if $g(z) > 0$, then $g(x) > 0$ for each $x \in (z,c)$ and $x^+(n,h,z)$ converges to c, if $g(z) < 0$, then $g(x) < 0$ for each $x \in (z,c)$ and $x^-(n,h,z)$ converges to c which completes the proof. □

When $a < z < 0$, using Lemma 2, some positive root c can also be found provided $z < c < -z$. In our algorithm, in two different steps we find negative and positive roots. Therefore, when we are finding negative roots we shall omit the the positive roots in the process and vice versa. Otherwise, some roots are calculated twice that is not desirable. Algorithm 3 Line 22 shows that when we are looking for negative roots and we reach a positive root we stop the process. In Algorithm 4, Line 5 we do the same when we are finding positive roots.

We can apply Lemma 2 to find the roots of a polynomial as follows.

Corollary 1. *Suppose $g(x) = a_m x^m + \cdots + a_1 x + a_0$ for each $x \in [a,b]$ be a polynomial with real coefficients, and define $f(x) = m|a_m|x^{m-1} + \cdots + 2|a_2|x + |a_1| + 1$ for each $x \in \mathbb{R}$, $h(x) = \frac{g(x)}{f(|z|)}$ for each $x \in [a,b]$. Assume $a \leq z < 0$ and c be the nearest root of g to z in this interval such that $c \in [z,-z]$. Then, one of the sequences $\{x^+(n,h,z)\}$ or $\{x^-(n,h,z)\}$ converges to c. The result holds if we replace the condition $a < z < 0$ with $0 < z < b$.*

Proof. We have $g^{'}(x) \leq f(|z|)$ for each $x \in [-z,z]$, and by Lemma 2 the result follows. □

Now we explain an algorithm that finds the roots of a polynomial or a differentiable function $g(x)$ that is dominated by a polynomial in the interval $[a,b]$. We find these roots in two steps as follows.

- (Case 1: Finding the negative roots) Suppose that $a < c < 0 < b$ where c is the nearest root of g to the right hand side of a. We start with a and define the sequences $x_+(n,h,z)$ and $x_-(n,h,z)$ as in Lemma 2. One of these sequences converges to c when $z = a$. We define the forward-limit function as before (see Line 8 of Algorithm 3). We have already given very detailed explanation of this function. The only difference here is that, when $a > 0$, the forward-limit function can not be applied to return the nearest root of g at the right side of a. To see this, suppose $a > 0$ and we start

from a. Without loss of generality, assume $g(a) > 0$. The interval at which the sequence $x_+(n,a)$ is convergent is $(-a,a)$. Since $x_+(n,h,a) \geq a$ for each $n \geq 0$, so $x_+(n,h,a) \notin (-a,a)$ and we have no guarantee of convergence. Therefore, we use forward-limit function just for finding negative roots of g.

- (Case 2: Finding the positive roots) Suppose that $a < 0 < c < b$ where c is the nearest root of g to the left hand side of b. This time we start from $z = b$ and use the backward-limit function defined in Line 17 of Algorithm 3 to find c. Similarly, the back-ward limit function also does not work when $b < 0$.

Now, we find all roots of g in $[a,b]$ as follows. If $a < 0$, through a while loop first we find all negative roots z with $a \leq z \leq 0$ as described in *Case*1 (see Line 18 of Algorithm 4). If $b \geq 0$, through a while loop we add all positive roots z with $0 \leq z \leq b$ as described in *Case*2 (see Line 1 of Algorithm 4). These two while loops will find all the roots of g in $[a,b]$. We notice that, if $z = 0$ be a root of g, then we will have two estimations for this root.

Suppose we are finding negative roots of g and we start with the initial point $z < 0$. After some iterations and replacing z with $y^+(z)$ inside the forward-limit function we may have $z > 0$. This shows that we have crossed the origin point $z = 0$. As we know, our root-finding process is no longer able to function and we should stop. We have the same situation when we are finding positive roots. So, in these cases, we add a condition to stop the while loop (see Line 11 of Algorithm 3). Now, Algorithms 3, 4 find all the roots of a polynomial or a differentiable function that is dominated by a polynomial on a finite closed interval. See Example 4 to know how we apply this method to find the roots of a differentiable function that is dominated by a polynomial on an interval.

4 Rate of convergence and stability

Generally, it is not always possible to transform a sequence into a new one to accelerate the rate of its convergence [11]. However, by Theorem 5, the rate of convergence of a linearly convergence sequence can be accelerated. In this part, we show that for any differentiable function g that has no roots of multiplicity greater than 1, our sequences always converge linearly to the roots of this function. To see some other results for increasing the rate of convergence we refer the reader to [13], [16], [6].
The convergence of our sequences are stable. Suppose that x_0 is a starting point for the sequences $\{x_n^+\}$ and $f(x_0) > x_0$. By Lemma 1, $\{x^+(n,x_0)\}$ converges to c, where c is the nearest root of g in $[x_0,+\infty)$. Since f is a continuous function, there exists a $\delta > 0$ such that $f(z_0) > z_0$ for each $z_0 \in (x_0-\delta, x_0+\delta)$. Therefore, by Lemma 1, $\{x^+(n,z_0)\}$ converge to c for each $z_0 \in (x_0-\delta, x_0+\delta)$. This shows that a small change of the starting point x_0 does not change the result. Therefore, our method is stable.

Lemma 3. *Let $g : [a,b] \to \mathbb{R}$ be a differentiable function, c be a fixed point of g such that g' is continuous at c. If $x^+(n,a) \to c$, then $g'(c) \leq 0$ and the convergence is linear. If $x^-(n,a) \to c$, then $g'(c) \geq 0$ and the convergence is again linear. If $g'(c) = 0$ and either $x^+(n,a) \to c$ or $x^-(n,a) \to c$, then the convergence is sublinear.*

Proof. We denote $x^+(n,a)$ simply by $\{x_n^+\}$ for each $n \geq 0$. By Lemma 1, $\{x_n^+\}$ is a monotone sequence. Suppose that $\{x_n^+\}$ is an increasing sequence. For each $n \geq 0$ there exists $x_n^+ \leq \theta_n \leq c$ such that,

$$\frac{x_{n+1}^+ - x_n^+}{x_n^+ - c} = \frac{g(x_n^+)}{2(x_n^+ - c)} = \frac{g(x_n^+) - g(c)}{2(x_n^+ - c)} = \frac{g'(\theta_n)}{2}.$$

Thus

$$\frac{|x_{n+1}^{+}-c|}{|x_n^{+}-c|}=1+\frac{g^{'}(\theta_n)}{2}. \tag{8}$$

We have $\theta_n \to c$ as $n \to \infty$. Define $\sigma_n = |x_n^{+}-c|$ for each $n \geq 0$. Since $g^{'}$ is continuous at c, by taking limit of both sides of Equation (8) and the fact that $\theta_n \to c$ as $n \to \infty$, we get

$$\lambda=\lim_{n\to\infty}\frac{\sigma_{n+1}}{\sigma_n}=1+\frac{g^{'}(c)}{2}. \tag{9}$$

From Equation (9) we deduce that $g^{'}(c) \leq 0$. By Definition 4 we have the following cases:

- if $g^{'}(c)=0$, then $\lambda=1$ and the sequence $\{x_n^{+}\}$ converges sublinearly to c.
- if $g^{'}(c)<0$, then $0 \leq \lambda < 1$ and the sequence $\{x_n^{+}\}$ converges linearly to c.

When $\{x_n^{+}\}$ is a decreasing sequence, the proof is similar. The proof for the sequence $\{x^{-}(n,a)\}$ follows similarly. □

We notice that, when g is a polynomial and c is a root of g with multiplicity greater than 1, then $g^{'}(c)=0$, and these sequences converge sublinearly to c.

5 Examples and pseudo codes

The following example that is from Khandani et al. [17] shows that the condition $t \leq \frac{1}{1+L}$ is necessary for the convergence of the sequence $\{x_n\}$ in Proposition 1.

Example 1. Define $g:[0,1]\to[0,1]$ as

$$g(x)=\begin{cases}1, & 0\leq x\leq \frac{3}{8},\\ -2x+\frac{7}{4}, & \frac{3}{8}\leq x\leq \frac{7}{12},\\ -6x+\frac{49}{12}, & \frac{7}{12}\leq x\leq \frac{49}{72},\\ 0, & \frac{49}{72}\leq x\leq 1.\end{cases}$$

Function g is a continuous 8-Lipschitz mapping and $c=\frac{7}{12}$ is the unique fixed point of it. Let $x_0=\frac{23}{63}$ and for each $n\geq 1$ define:

$$x_{n+1}=(1-t)x_n+tg(x_n), \quad t\in(0,1]. \tag{10}$$

If $t=\frac{1}{2}$, then $x_1=\frac{86}{126}, x_2=\frac{43}{126}, x_3=\frac{169}{252}, x_4=\frac{23}{63}=x_0$. This shows that $\{x_n\}$ is not a convergent sequence. Since $t=\frac{1}{2}$ and $L=8$, the condition $t\leq\frac{1}{1+L}$ does not hold. We notice that g satisfies in all other conditions of Proposition 1. Therefore, the condition $t\leq\frac{1}{1+L}$ is necessary for the convergence of $\{x_n\}$.

Example 2. This example uses Algorithms 1 and 2 to find the roots of a given function and reports about the missing roots. Define $h(0)=0$ and $h(x)=x^2 sin(\frac{1}{x})$ for each $x\neq 0$. Function h is a differentiable on [0,1] and $h^{'}(x)\leq 3$ for each $x\in[0,1]$. Therefore the function $g=2h/3$ satisfies all conditions of Theorems 3 and 4 on the interval $[0,1]$ where $a=.00001, b=1$. We want all roots to be calculated

with 16 decimal precision, so set $\delta = 10^{-16}$. We already know all the roots of g in this interval. The set of all these roots is $A = \{x_k = \frac{1}{k\pi} : 10^{-5} \leq \frac{1}{k\pi} \leq 1\}$. The number of these roots is 31830. We run Algorithm 2 for different ε until no missed root is reported (see Table1). All roots are found when $\varepsilon = .0000000001$. Comparing these roots with the array $[1/(\pi), 1/(2\pi), 1/(3\pi), \ldots]$, we see that all found roots have been estimated with precision 10^{-16}. We notice that the roots have been found from the left to the right hand side of this interval. After reversing the order of the found roots, we see that the calculated root for $n = 20000$ is $1.591469857426082 \times 10^{-5}$, the value of this root with 16 decimal precision is: $1/(20001\pi) = 1.591469857426082 \times 10^{-5}$. Table 1 shows that how our algorithm reports about the missing roots on the interval $[.00001, 1]$ as ε is getting smaller.

Table 1: The number of found and missed roots for $g(x) = x^2 \sin(\frac{1}{x})$ in the interval $[.00001, 1]$ for different ε.

ε	All roots	Missed roots	The number of found roots
.1	31830	Yes	3
.01	31830	Yes	10
.001	31830	Yes	33
.00001	31830	Yes	330
.0000001	31830	Yes	3218
.000000001	31830	Yes	23658
.0000000001	31830	No	31830

Source: Python 3.9.7 (tags/v3.9.7:1016ef3) [MSC v.1929 32 bit (Intel)] on win32

Example 3. To find the roots of a given function $g : [a,b] \to \mathbb{R}$ by Newton-Raphson method, the starting points are chosen as follows. Suppose $n > 0$, define $A = \{a + \frac{i(b-a)}{n}, 0 \leq i \leq n\}$. At each starting point $x \in A$, Newton-Raphson method is applied. When n is large enough, each root is near to a starting point. Then, the Newton-Raphson method at that starting point converges to that root. Therefore, by increasing n all roots of g in this interval are found. This is the usual way of choosing a set of starting points for Newton-Raphson method. We notice that:

- many starting points converge to the same root.
- for some starting points, Newton-Raphson method does not converge to a root.
- for some starting points, Newton-Raphson method converges to a root out of the interval [a,b].

Therefore, a considerable amount of time will be wasted when we apply Newton-Raphson method. We calculate the wasted time as follows. Suppose we have found all roots of g in $[a,b]$ where A is the set of starting points defined above. Suppose f is the number of all these roots. We remove the repeated roots and roots outside of the interval $[a,b]$ and suppose u is the remained number of roots. Then, the ratio $\frac{u}{f}$ is the relative efficient time that has been spent on the calculation of the roots. The $1 - \frac{u}{f}$ is called the wasted time. We have applied our method and Newton-Raphson method to estimate the roots of the function presented in Example 2. Table 2 shows that over 98 percent of the time is wasted when we apply Newton-Raphson method. Table 2 shows that our method needs less time than Newton-Raphson method to find all the roots of g. It is worth mentioning that in computing the speed of both of these methods,

the time spent by printing the results to the screen has not been taken into account. The pseudo-code for applying our method to find the roots of this function has been presented in Algorithm 1. Algorithm 5 calculates the wasted time of the Newton method. To calculate the limit of the sequence $x(n,a)$ in Line 6 of Algorithm 5, we refer the reader to [19], page 331.

Table 2: A comparison between Newton-Raphson and our method. We calculate the time needed to find the roots of the function $g(x) = x^2 \sin(\frac{1}{x})$ in the interval $[a,b]$ where $a = .01, .001, .0001$ and $b = 1$ for these two methods. In this table $m = |A|$ where A is the set of starting points, FR is the number of found roots and AR is the number of all the roots in $[a,b]$. We increase m until we find all roots, equivalently when $FR = AR$. The NR and OM denote Newton-Raphson and our method respectively, WT and ET denote the related wasted and efficient time of these methods, respectively.

Method	a	m	FR	Time	AR	WT	ET
NR	.01	2000	28	0.0624	31	0.986	0.0140
NR	.01	3000	31	.1093	31	.9896	0.0103
OM	.01	-	31	0.0624	31	0.0	1
NR	.001	272×10^3	311	8.4372	318	0.9988	0.0011
NR	.001	274×10^3	318	8.4685	318	0.9988	0.0011
OM	.001	-	31	0.2540	318	0.0	1
NR	.0001	25×10^5	1560	77.5290	3183	0.9993	0.0007
NR	.0001	95×10^5	2519	294.1010	3183	0.9997	0.0002
OM	.0001	-	3183	1.2031	3183	0.0	1
NR	.00001	pow(10,7)	3586	298.5306	31830	.9996	0.0003
OM	.00001	-	31830	8.6601	31830	0.0	1

Source: Python 3.9.7 (tags/v3.9.7:1016ef3) [MSC v.1929 32 bit (Intel)] on win32

Example 4. Define $g(x) = x^7 \sin(x) - x^5 \cos(x) + x + 1$ for each $x \in \mathbb{R}$, we want to find the roots of this function on each interval $[-10,10]$. For each $x \in \mathbb{R}$ we have:

$$|g^{'}(x)| \leq x^7 + 7x^6 + x^5 + 5x^4 + 1. \tag{11}$$

Define $p(x) = x^7 + 7x^6 + x^5 + 5x^4 + 1$ for each $x \in \mathbb{R}$ which dominates the function $g^{'}$. Now, by Lemma 2 we can run Algorithms 3 and 4 with the precision $\delta = 10^{-16}$ to find the roots of g on the interval $[-10,10]$ that are as follows

$$-9.413492235971914, -6.257667541802755, -3.0324128980671126,$$
$$9.4360101786797, 6.308290722466052, 3.2378237299099184.$$

The following example shows that our method introduced in Section 3 can be applied to polynomials with large degrees.

Example 5. Define $g(x) = x^{75} - 3x^{50} + x^{25} - 2$ for each $x \in \mathbb{R}$, g has just one real root which is in $[0,2]$ and the 74 other roots are complex [21]. By putting $a = -10000, b = 10000$, this root is estimated as $c = 1.0434116316793722$. This example uses Algorithm 3 to find the roots of a polynomial in a finite interval.

Algorithm 1 This algorithm finds the roots of a function on a closed interval.

Require: $\delta, \varepsilon > 0, a < b, a + \varepsilon < b$, $g : [a,b] \to \mathbb{R}$ is differentiable and $|g^{'}| \leq 2$ on $[a,b]$.

$F = \{\}$
Function $y^{+}(z)$
return $z + sign(g(z))\frac{g(z)}{2}$
end Function
Function $y^{-}(z)$
return $z - sign(g(z))\frac{g(z)}{2}$
end Function
Function $f\_limit(z)$
while $|g(z)| > \delta$ and $z \leq b$ **do**
    $z \leftarrow y^{+}(z)$
    if $y^{+}(z) = z$ **then**
        Break
    end if
end while
return z
end Function
In the definition of f_limit function replacing y^{+} with y^{-}, the b_limit function is defined.
$z \leftarrow a$
while $z \leq b$ **do**
    c=$f\_limit(z)$
    if $c > b$ **then**
        Break
    else
        $F \leftarrow F \cup \{c\}, z \leftarrow c + \varepsilon$
    end if
end while
if $F = \phi$ **then**
    g has no roots in $[a,b]$.
else
    F is the set of all roots of g in $[a,b]$ found for this ε.
end if

Algorithm 2 This algorithm can be added to Algorithm 1 and makes a report about the missing roots.

$lost\_roots \leftarrow 0$
Function missed_root($root$)
$z \leftarrow root + \varepsilon$
if $b\_limit(z) \neq root$ **then**
  return 1
else
  return 0
end if
end Function
$z \leftarrow a$
while $z \leq b$ **do**
  if $f\_limit(z) > b$ **then**
    Break
  else
    $c \leftarrow f\_limit(z)$, F$\leftarrow F \cup \{c\}$, $z \leftarrow c + \varepsilon$
    lost_roots$\leftarrow$ missed_root(c)+ lost_roots
  end if
end while
if $lost\_roots = 0$ **then**
  if $F = \phi$ **then**
    There are no roots in $[a,b]$.
  else
    All the roots has been fond in $[a,b]$.
  end if
else
  Some roots are missed.
end if

Algorithm 3 Estimating the roots of differentiable functions dominated by a polynomial on an interval.

Require: $a,b \in \mathbb{R}$ with $a<b$, $n \in \mathbb{N}$, $p(x)=a_n x^n+\cdots+a_1x+a_0$ is a polynomial and $|g(x)| \le p(x)$ for each $x \in [a,b]$.

$F=\{\}$, lost-roots = 0, $f(x)=n|a_n|x^{n-1}+\cdots+|a_1|x+|a_0|$
Function $y^+(z)$
return $z+sign(g(z))\frac{g(z)}{f(|z|)}$
end Function
Function $y^-(z)$
return $z-sign(g(z))\frac{g(z)}{f(|z|)}$
end Function
Function $f\_limit(z)$
while $|g(z)|>\delta$ **do**
  $z \leftarrow y^+(z)$
  if $y^+(z)=z$ or $z>0$ **then**
    Break
  end if
end while
Return z
end Function
b_limit function: In the definition of f_lim function replace $y^+(z)$ and $z>0$ with $y^-(z)$ and $z<0$ respectively.
if $a<0$ **then**
  $z \leftarrow a$
  while $z \le 0$ **do**
    $c \leftarrow f\_limit(z)$
    if $c>0$ **then**
      Break
    else
      if $c \le b$ **then**
        F$\leftarrow F \cup \{c\}$, $z \leftarrow c+\varepsilon$
      else
        Break
      end if
    end if
  end while
end if

Algorithm 4 This is the continuation of Algorithm 3 from Line 32.

if $b > 0$ **then**
  $z \leftarrow b$
  while $z \geq 0$ **do**
    $c \leftarrow b\_limit(z)$
    if $c < 0$ **then**
      Break
    else
      if $c \geq a$ **then**
        F$\leftarrow F \cup \{c\}$, $z \leftarrow c - \varepsilon$
      else
        Break
      end if
    end if
  end while
end if
The set of found roots is F

Algorithm 5 Estimating of the time wasted by Newton-Raphson method.

Require: A is the set of starting points, F is an empty list. tol=exp(10,-16), $[a,b]$ is an interval, $g : [a,b] \to \mathbb{R}$ is a differentiable function.

$x(0,a) = a$ for each $a \in \mathbb{R}$.
for $n = 0, 1, \ldots$ **do**
  $x(n+1,a) \leftarrow x(n,a) - \frac{g(x(n,a))}{g'(x(n,a))}$
end for
for $a \in A$ **do**
  $F.add(lim_{n\to\infty} x(n,a))$
end for
$f \leftarrow length(F)$//the number of all roots found, applying Newton-Raphson method to A.
Each member of F is round down as follows.
for $x \in F$ **do**
  $x \leftarrow floor(x, 15)$
end for
unique_list=[ ]
//We remove the repeated roots, roots out of $[a,b]$ or numbers that approximate the same root. So, we remove the roots that their first 15th decimals are equal.
for $x \in F$ **do**
  if $x \notin unique\_list \wedge a \leq x \leq b$ **then**
    $unique\_list.add(x)$
  end if
end for
$u \leftarrow$length($unique\_list$)
$wasted\_time \leftarrow 1 - \frac{u}{f}$

6 Conclusions

In this paper, we introduced some new root-finding algorithms. Through a concrete example, we showed that these new methods spend less time than Newton-Raphson method to find all roots of a differentiable function in a closed interval. Our root-finding method for differentiable functions is new and can be applied to polynomials as well. We showed that the rate of convergence of our method can be increased. Also, this method does not rely on the isolation process of the roots that is very time consuming. All these facts show that this method can be regarded as a useful tool in finding the real roots of polynomials.

References

[1] G. Alefeld, F. Potra, Y. Shi, *On enclosing simple roots of nonlinear equations*, Math. Comput. **61** (1993) 733–744.

[2] D. Bailey, *Krasnoselski's theorem on the real line*, Amer. Math. Monthly **81** (1974) 506–507.

[3] D. Borwein, J. Borwein, *Fixed point iterations for real functions*, J. Math. Anal. Appl. **157** (1991) 112–126.

[4] R.P. Brent, *An algorithm with guaranteed convergence for finding a zero of a function*, Comput. J. **14** (1971) 422–425.

[5] C. Brezinski, *Algorithmes d'Accélération de la Convergence-Étude Numérique*, Editions Technip, Paris, 1978.

[6] C. Brezinski, M.R. Zaglia, *Generalizations of aitken's process for accelerating the convergence of sequences*, Comput. Appl. Math. **26** (2007) 171–189.

[7] T.R. Chandrupatla, *A new hybrid quadratic/bisection algorithm for finding the zero of a nonlinear function without using derivatives*, Adv. Eng. Softw. **28** (1977), 145–149.

[8] C. Chidume, *Some Geometric Properties of Banach Spaces*, Springer, London, 2009.

[9] G.E. Collins, A.G. Akritas, *Polynomial real root isolation using descarte's rule of signs*, SYMSAC'76: Proceedings of the third ACM symposium on Symbolic and algebraic computation, (1976) 272–275.

[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, 2009.

[11] J. Delahaye, B. Germain-Bonne, *Résultats négatifs en accélération de la convergence*, Numer. Math. **35** (1980) 443–457.

[12] J.F. Epperson, *An Introduction to Numerical Methods and Analysis*, John Wiley & Sons, 2021.

[13] N. Gattal, A. Chibi, *An improvement of steffensen's method for solving nonlinear equations*, Glob. J. Pure. Appl. Math. **12** (2016), 935–941.

[14] P. Henrici, *Elements of Numerical Analysis*, Tech. Rep., John Wiley & Sons, 1964.

[15] B.P. Hillam, *A generalization of Krasnoselski's theorem on the real line*, Math. Mag. **48** (1975) 167-168.

[16] P. Jain, K. Sethi, *Aitken type methods with high efficiency*, Trans. A. Razmadze Math. Inst. **172** (2018) 223–237.

[17] H. Khandani, F. Khojasteh, *An iterative method for estimation the roots of real-valued functions*, Sahand. Commun. Math. Anal. **20** (2023) 95–106.

[18] M. Krasnoselskii, *Two observations about the method of successive approximations uspehi math*, Appl. Math. Comput. **10** (1955) 123–127.

[19] Q. Kong, T. Siauw, A. Bayen, *Python Programming and Numerical Methods: A Guide for Engineers and Scientists*, Academic Press, 2020.

[20] J.M. McNamee, V.Y. Pan, *Numerical Methods for Roots of Polynomials-Part II*, Academic Press, 2013.

[21] R. Oftadeh, M. Nikkhah-Bahrami & A. Najafi-A, *A novel cubically convergent iterative method for computing complex roots of nonlinear equations*, Appl. Math. Comput. **217** (2010) 2608–2618.

[22] V.Y. Pan, *Solving a polynomial equation: some history and recent progress*, SIAM Rev. **39** (1977) 187–220.

[23] C. Ridders, *A new algorithm for computing a single root of a real continuous function*, IEEE. Trans. Circuits Syst. **26** (1979) 979–980.

[24] J.M. Rojas, *Book Review: Algorithms in Real Algebraic Geometry by S. Basu, R. Pollack, and M.F. Roy*, Springer, 2007.

[25] E. Süli, D.F. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, Cambridge, 2003.

[26] A.J.H. Vincent, *Note sur la résolution des équations numériques*, J. Math. Pures Appl. **1** (1834) 341–372.