Received 17 July 2022
Accepted 26 May 2023

DOI: 10.52547/CMCMA.1.2.74
AMS Subject Classification: 41A05; 34G20; 65H04

A new approach for solving nonlinear equations

Marzieh Dehghani-Madiseh[a]

In this paper, we describe and analyze an efficient method to find the roots of a general one variable function $f : \mathbb{R} \to \mathbb{R}$. The proposed method is based on partitioning an interval (that probably contains root(s) of f) into subintervals. From this point of view, we name this method a finite element approach for root finding. Also the convergence analysis of the presented method is presented. The new approach can be generalized to estimate the roots of the multivariable functions in higher dimensions. Also it is capable to find all of the roots of the function on a determined interval. Finally, numerical examples are given to illustrate the effectiveness of the new method. Copyright © 2022 Shahid Beheshti University.

Keywords: Nonlinear function; finite element approach; piecewise Lagrange interpolation.

1. Introduction

Finding the zeros of a nonlinear equation, is a classical problem which has nice applications in various branches of science and engineering. Since the zeroes of a function generally cannot be computed exactly, rootfinding algorithms are provided to approximate the zeroes of the function. Consider the general nonlinear function $f : \mathbb{R} \to \mathbb{R}$. The roots of this function are the values of x for which

$$f(x) = 0.$$

So, for example $x^* = 0$ is a root of $f(x) = x$, $f(x) = x^2$, $f(x) = \sqrt{x}$, and $f(x) = \ln(x + 1)$.

Nonlinear equations arise in all branches of science, engineering, and technology. In recent years, a large number of articles for root finding have been appeared in the literature. Iterative methods for finding a simple real root x^* of a nonlinear equation $f(x) = 0$ have good convergence properties if the initial approximation is reasonably close to the root. But choosing a good initial approximation is very difficult, specially when the function f has improper behavior. One of the most popular iterative methods is the Newton method which has a locally quadratically order of convergence. To improve the local order of its convergence, many modified methods have been proposed [27, 10, 9, 23, 24]. One of them is Ostrowski's method which is a fourth order method and requires two evaluations of the function and one evaluation of its first derivative. But all of these methods are iterative methods and for a good convergence, must have a good initial approximation, also all of them can only find at most one of the roots of the function f. Many authors proposed numerical methods for solving this equation such as the bisection method, the chord method, the secant method, the false position (or regula falsi) method, the Möller's method, the modified regula falsi method, the Pegasus method, and so on, see [26, 14, 15, 3, 32, 1, 36, 25, 28, 12, 13, 2, 22, 29, 4, 31, 17, 34, 35, 5, 19, 21].

In this paper, we want to use a finite element approach to find the roots of a nonlinear equation. For this purpose, for $f : [a, b] \to \mathbb{R}$, we introduce a partition T_h of $[a, b]$ into K subintervals $I_j = [x_j, x_{j+1}]$ of length h_j, with $h = \max_{0\leq j\leq K-1} h_j$ and then employ Lagrange interpolation on each I_j using $k + 1$ equally spaced nodes $\{x_j^{(i)}, 0 \leq i \leq k\}$.

For $k \geq 1$, we introduce the piecewise polynomial space

$$\chi_h^k = \{v \in C^0[a, b] : v|_{I_j} \in \mathbb{P}_k(I_j), \forall I_j \in T_h\},$$

which is the space of the continuous functions over $[a, b]$ whose restrictions on each I_j are polynomials of degree less than or equals to k. Then, for any continuous function f in $[a, b]$, the piecewise interpolation polynomial $\Pi_h^k f$ coincides on each I_j with

[a] *Department of Mathematics, Faculty of Mathematical Sciences and Computer, Shahid Chamran University of Ahvaz, Ahvaz, Iran. Email: m.dehghani@scu.ac.ir.*
[*] *Correspondence to: M. Dehghani-Madiseh.*

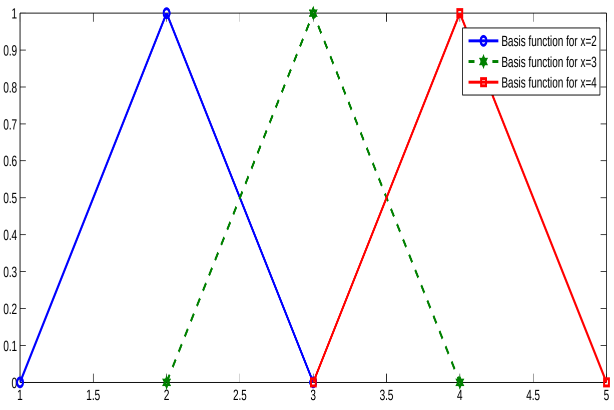

Figure 1. Basis functions corresponding to the $x = 2$, $x = 3$, $x = 4$ as main nodes in interval $[1, 5]$

the interpolating polynomial of $f|_{I_j}$ at the $k + 1$ nodes $\{x_j^{(i)}, 0 \le i \le k\}$. As said in [30], if $f \in C^{k+1}([a, b])$ using the interpolation error within each interval we obtain the following error estimate

$$\|f - \Pi_h^k f\|_\infty \le C h^{k+1} \|f^{(k+1)}\|_\infty, \tag{1}$$

where $\|.\|_\infty$ is a norm defined by $\|f\|_\infty = \sup_{x\in[a,b]} |f(x)|$. Note that a small interpolation error can be obtained even for low k provided that h is sufficiently small.

Now the new method is based on the root finding of the piecewise interpolation polynomial $\Pi_h^k f$ on each subinterval I_j, $j = 0, 1, ..., K - 1$. Due to the convergence of this piecewise interpolation polynomial to the function f, we can show that the approximation roots that are found from the new method are convergent to the exact roots of the function f. Our approach is based on performing the following steps:

Step 1: Divide the interval $[a, b]$ into K subintervals $I_j = [x_j, x_{j+1}]$.
Step 2: Form the piecewise interpolation polynomial $\Pi_h^k f$.
Step 3: Find the roots of the interpolation polynomials on each subinterval I_j.
Step 4: For each root, test for $f(root)$, if it is sufficiently small, stop. Otherwise, refine the mesh size or degree of the piecewise interpolation polynomial.

If $k = 1$ then we have the piecewise linear Lagrange interpolation which by dividing the interval $[a, b]$ into K subintervals $I_j = [x_j, x_{j+1}]$ is of the form

$$\Pi_h^1 f = \sum_{j=0}^{K} c_j \beta_j(x),$$

where $c_j = f(x_j)$ and $\beta_j(x)$ is the basis function corresponding to the jth node, i.e.,

$$\beta_j(x) = \begin{cases} \frac{x-x_{j-1}}{x_j-x_{j-1}} & x \in [x_{j-1}, x_j], \\ \frac{x-x_{j+1}}{x_j-x_{j+1}} & x \in [x_j, x_{j+1}], \\ 0 & \text{otherwise.} \end{cases}$$

In Figure 1, the basis functions corresponding to the main nodes $x = 2, 3, 4$ on interval $[1, 5]$ are displayed.
The linear interpolation polynomial $\Pi_h^{1,j} f$ on subinterval $[x_j, x_{j+1}]$, $j = 0, 1, .., K - 1$, is

$$\Pi_h^{1,j} f(x) = f(x_j)\frac{x - x_{j+1}}{x_j - x_{j+1}} + f(x_{j+1})\frac{x - x_j}{x_{j+1} - x_j}.$$

The root of this polynomial is $x = \frac{f(x_{j+1})x_j - f(x_j)x_{j+1}}{f(x_{j+1}) - f(x_j)}$ which if it belongs to $[x_j, x_{j+1}]$ then we consider it as an approximate root of f. In Figure 2, we show graphs of the function $f(x) = \sin x$ and its piecewise linear Lagrange interpolation on $[0.5, 4]$. We divide this interval into five equally spaced subintervals. As can be seen, both graphs have a root on the interval $[3, 3.5]$.

Now, for $k = 2$ we divide interval $[a, b]$ into K subintervals $I_j = [x_j, x_{j+1}]$, $j = 0, 1, ..., K - 1$, of length h_j, and divide the jth subinterval as $[x_j, x_{j+1}] = [x_j, x_{j+\frac{1}{2}}] \cup [x_{j+\frac{1}{2}}, x_{j+1}]$ with $x_{j+\frac{1}{2}} = x_j + \frac{h_j}{2}$. The piecewise quadratic Lagrange interpolation is of the

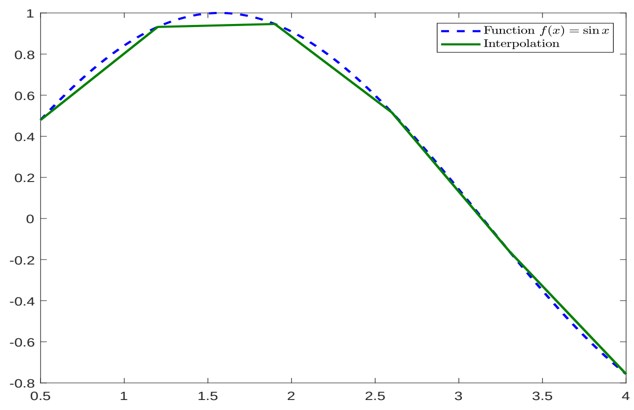

Figure 2. Function $f(x) = \sin x$ and its piecewise linear Lagrange interpolation

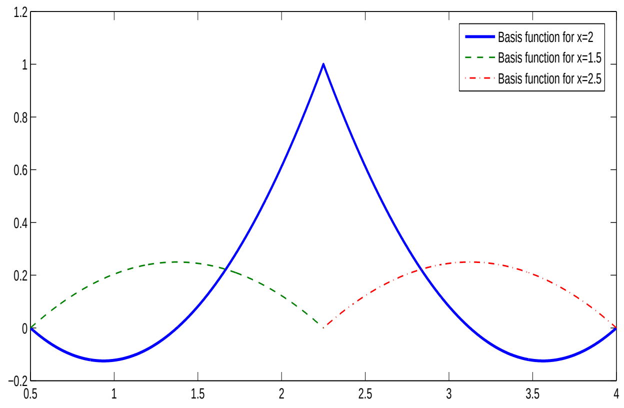

Figure 3. Basis functions corresponding to $x = 2$ as a main node in interval $[1, 3]$ and corresponding to $x = 1.5$ and $x = 2.5$ as two middle points of intervals $[1, 2]$ and $[2, 3]$

form

$$\Pi_h^2 f = \sum_{j=0}^{K} c_j \beta_j(x) + \sum_{j=1}^{K} c_{j-\frac{1}{2}} \beta_{j-\frac{1}{2}}(x),$$

therein $c_j = f(x_j)$ and $c_{j-\frac{1}{2}} = f(x_{j-\frac{1}{2}})$. $\beta_j(x)$ and $\beta_{j-\frac{1}{2}}(x)$ are the basis functions corresponding to the jth and $(j-\frac{1}{2})$th nodes, respectively. $\beta_j(x)$ is

$$\beta_j(x) = \begin{cases} \frac{x-x_{j-1}}{x_j-x_{j-1}} \cdot \frac{x-x_{j-\frac{1}{2}}}{x_j-x_{j-\frac{1}{2}}} & x \in [x_{j-1}, x_j], \\ \frac{x-x_{j+\frac{1}{2}}}{x_j-x_{j+\frac{1}{2}}} \cdot \frac{x-x_{j+1}}{x_j-x_{j+1}} & x \in [x_j, x_{j+1}], \\ 0 & \text{otherwise.} \end{cases}$$

Note that two polynomials in definition of $\beta_j(x)$ are named the shape functions corresponding to node x_j. Also we have

$$\beta_{j-\frac{1}{2}}(x) = \begin{cases} \frac{x-x_{j-1}}{x_{j-\frac{1}{2}}-x_{j-1}} \cdot \frac{x-x_j}{x_{j-1}-x_j} & x \in [x_{j-1}, x_j], \\ 0 & \text{otherwise.} \end{cases}$$

In Figure 3, one can see the basis functions corresponding to $x = 2$ as a main node in the interval $[1, 3]$ and corresponding to $x = 1.5$ and $x = 2.5$ as two middle points of intervals $[1, 2]$ and $[2, 3]$, respectively.

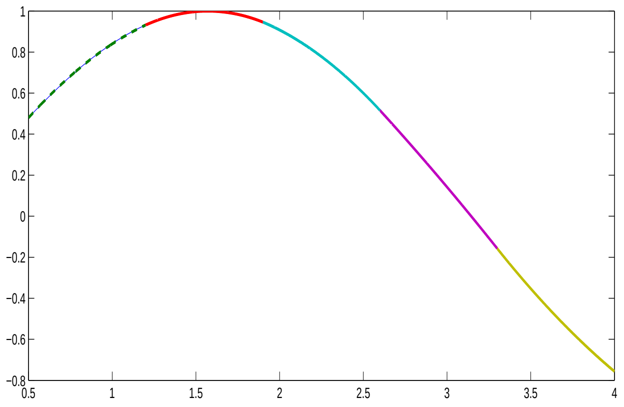

Figure 4. Function $f(x) = \sin x$ '-', and its piecewise quadratic Lagrange interpolation '- - -'

The quadratic interpolation polynomial $\Pi_h^{2,j} f$ on interval $[x_j, x_{j+1}]$, $j = 0, 1, ..., K-1$, is

$$\Pi_h^{2,j} f(x) = f(x_j)\frac{x - x_{j+\frac{1}{2}}}{x_j - x_{j+\frac{1}{2}}} \cdot \frac{x - x_{j+1}}{x_j - x_{j+1}} + f(x_{j+\frac{1}{2}})\frac{x - x_j}{x_{j+\frac{1}{2}} - x_j} \cdot \frac{x - x_{j+1}}{x_{j+\frac{1}{2}} - x_{j+1}} + f(x_{j+1})\frac{x - x_j}{x_{j+1} - x_j} \cdot \frac{x - x_{j+\frac{1}{2}}}{x_{j+1} - x_{j+\frac{1}{2}}}.$$

If the real roots of this polynomial belong to the interval $[x_j, x_{j+1}]$ and $f(root)$ is sufficiently small, then we can consider them as approximate roots of function f. In Figure 4, we show graphs of the function $f(x) = \sin x$ and its piecewise quadratic Lagrange interpolation on the interval $[0.5, 4]$ which is divided into five equally spaced subintervals. Although, the mesh size is large but both graphs are coincident, which implies the good accuracy of the piecewise quadratic interpolation. It is obvious that with the same mesh size, the piecewise quadratic Lagrange interpolation finds a more accurate root for the function $f(x) = \sin x$ than the piecewise linear Lagrange interpolation. For other quantities of k we have the same discussion.

The rest of the paper is organized as follows: in Section 2, we study the convergence properties of the new method. Some suggestions and development of the method are briefly discussed in Section 3. Numerical experiments are presented in Section 4. Finally, in Section 5, we end the paper with brief concluding remarks.

2. Convergence analysis of the new method

In this section, we study the convergence analysis of the new approach. For this purpose, we introduce the following space

$$L^2(a,b) = \{f : (a,b) \to \mathbb{R} : \int_a^b |f(x)|^2 < \infty\}, \tag{2}$$

with

$$\|f\|_{L^2(a,b)} = (\int_a^b |f(x)|^2)^{\frac{1}{2}}. \tag{3}$$

Formula (3) defines a norm on $L^2(a,b)$. It is to be noted that we integrate from $|f|^2$ in the Lebesgue sense, see [33].

Theorem 1 *[30] Let $0 \le m \le k+1$, with $k \ge 1$ and assume that $f^{(m)} \in L^2(a,b)$ for $0 \le m \le k+1$, then there exists a positive constant C, independent of h, such that*

$$\|(f - \Pi_h^k f)^{(m)}\|_{L^2(a,b)} \le C h^{k+1-m} \|f^{(k+1)}\|_{L^2(a,b)}.$$

In particular, for $k = 1$, and $m = 0$ or $m = 1$, we obtain

$$\|f - \Pi_h^1 f\|_{L^2(a,b)} \le C_1 h^2 \|f''\|_{L^2(a,b)},$$

$$\|(f - \Pi_h^1 f)'\|_{L^2(a,b)} \le C_2 h \|f''\|_{L^2(a,b)},$$

for two suitable positive constants C_1 and C_2.

If function f satisfies in the conditions of Theorem 1 then the piecewise Lagrange interpolation polynomial $\Pi_h^k f$ converges to f. Therefore, we can conclude that the approximate roots that are found using the new method are convergent to the exact roots of the function f. Since if we get h sufficiently small, then the interpolating polynomial of $f|_{I_j}$ is nonincreasing or nondecreasing, therefore, if it has a root on the interval I_j, then we have

$$\Pi_h^k f(x_j)\Pi_h^k f(x_{j+1}) \leq 0,$$

and as we have

$$f(x_j) = \Pi_h^k f(x_j), \qquad f(x_{j+1}) = \Pi_h^k f(x_{j+1}),$$

then we conclude that $f(x_j)f(x_{j+1}) \leq 0$, and the intermediate value theorem implies that f has a root on this interval. By the above theorem, if h is sufficiently small, we will have higher accuracy in root finding.

With the above assumptions, it is obvious that if x_j^* is root of the function $\Pi_h^k f$ on the interval $[x_j, x_{j+1}]$, then the function f will have a root on this interval shown by x^* and at least we have $|x^* - x_j^*| < Ch$ wherein C is a constant. So, the new approach for is a convergent method.

One advantage of the new method is that it can find all roots of function f on the assumed interval $[a, b]$ while many well-known methods can only find one root of f on this interval. The second major benefit is that the method does not rely on the derivatives of the function. Thus it can be used to find roots of non-smooth functions. Also it can be generalized to the general n dimensional case and doesn't require the initial guess. But note that if we want to use large amounts of k for interpolating function f then we face with higher degree polynomials which cause some difficulties in root finding, so we suggest to choose k to be a small positive integer with sufficiently small amount of h.

3. Suggestions and development of the method

Since we apply the finite element technique for presenting the new method, we can use some properties of this technique to improve the new approach such as using the hierarchical approach for interpolation, refinement approaches (p-FEM, h-FEM, r-FEM, hp-FEM,...), adaptive approaches, and so on. In fact, for attaining methods with higher-order accuracy, we can refine the mesh or degree of the interpolating polynomial or relocate nodes. Also in the locations of the interval $[a, b]$ that are critical (for example consider the case where the function f has high oscillations or has zeros that are very close together), we can use adaptive approaches, i.e., use higher degrees for interpolating polynomials or smaller meshes on those locations.

On the other hand, we can use this approach for general n dimensional case. For example, for function $z = f(x, y)$ and finding set $\{(x, y) \in \mathbb{R}^2 : \ z = 0\}$, we can consider triangular or rectangular elements on the (x, y)-plane, and interpolate the function $z = f(x, y)$ on each element and then find roots of these piecewise interpolation polynomials. In this case, interpolation can be done with different degrees of polynomial or with different types of elements.

For an analysis of this idea, suppose $z = u(x, y)$ is a continuous function on $\Omega \subseteq \mathbb{R}^2$ where Ω is a bounded open domain with boundary Γ. Let $T_h = \{K\}$ be a triangulation of Ω. We define h to be the largest sides among all triangles K. Also, let $r \in \mathbb{R}$ be a non-negative natural number, $P_r(K)$ is the space of all polynomials of degree r on K and $C^0(K)$ is the space of all continuous functions on K. We define

$$D^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2}}, \tag{4}$$

therein $\alpha = (\alpha_1, \alpha_2)$ and α_i is a non-negative natural number with $|\alpha| = \alpha_1 + \alpha_2$. We now define two spaces $L_2(\Omega)$ and $H^k(\Omega)$ for $k = 1, 2, \ldots$, as

$$L_2(\Omega) = \{v : v \text{ is defined on } \Omega \text{ and } \int_\Omega |v|^2 dx < \infty\},$$

$$H^k(\Omega) = \{v \in L_2(\Omega) : D^\alpha v \in L_2(\Omega), \quad |\alpha| \leq k\},$$

and

$$\|u\|_{L_2(\Omega)} = \Big(\int_\Omega |v|^2 dx\Big)^{1/2},$$

$$|u|_{H^r(\Omega)} = \Big(\sum_{|\alpha|=r} \int_\Omega |D^\alpha u|^2 dx\Big)^{1/2}.$$

Theorem 2 *[20] Let $K \in T_h$ be a triangle with vertices a^i, $i = 1, 2, 3$. Given $u \in C^0(\Omega)$, let the interpolant $\Pi_h^r u \in P_r(K)$ be defined by*

$$\Pi_h^r u(a^i) = u(a^i), \qquad i = 1, 2, 3,$$

then we have the following estimation

$$\|u - \Pi_h^r u\|_{L_2(\Omega)} \leq Ch^{r+1} |u|_{H^{r+1}(\Omega)},$$

where $C > 0$ is a constant that is independent of h.

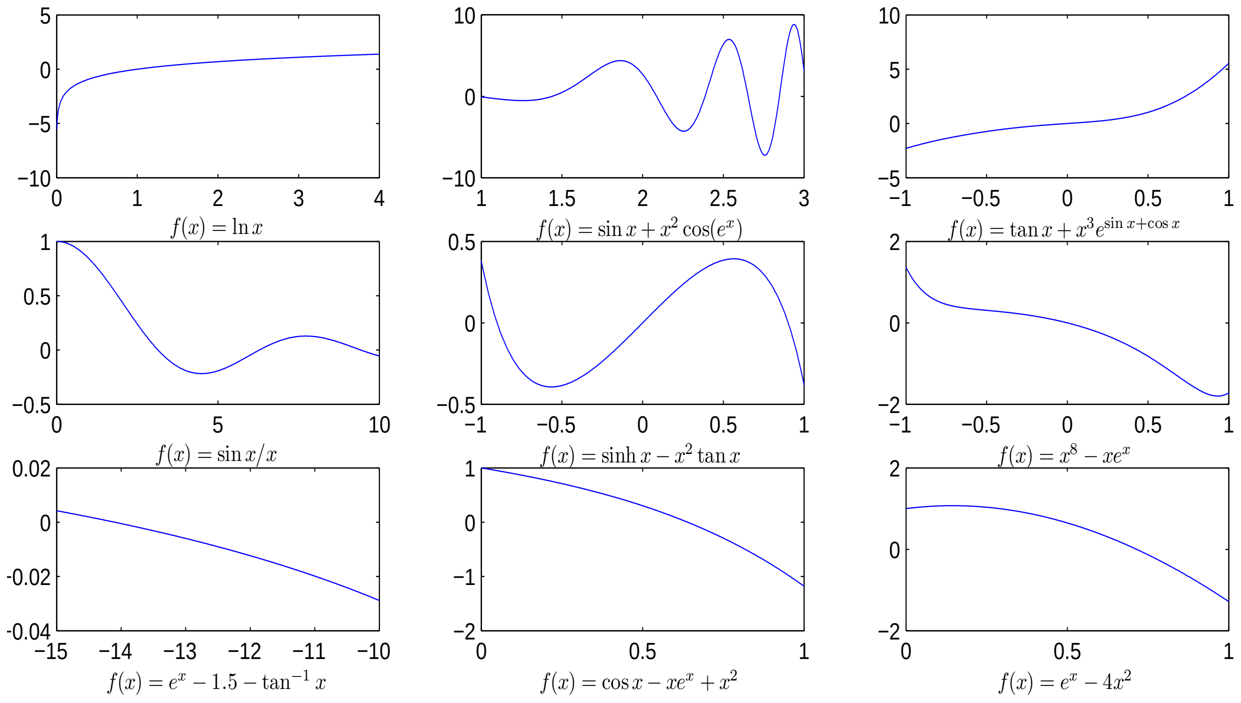

Figure 5. Graphs of the functions in nine examples

By the above theorem if we interpolate the function $u \in C^0(\Omega)$ on each element with piecewise polynomials, then roots of these polynomials on each element (if there exist) tend to the roots of the main function u. Of course, finding roots of a polynomial is very easier than a general function u.

For more details about finite element approaches, we refer the interested reader to [20, 11, 7, 18, 6, 8, 16].

4. Numerical experiments

In this section, we use some test problems to show the effectiveness of the new approach and compare the results for several mesh sizes and for piecewise linear, quadratic and cubic interpolating polynomials in terms of the executing time and quality of the obtained results. We arrange the obtained results in some tables therein the last column is for the executing times and all computational times are in seconds.

In the following, we present nine examples and show the graphs of the corresponding functions in Figure 5.

Example 1 *Consider*

$$f(x) = \ln(x).$$

By Figure 5, we find that this function has a root on $[0.5, 1.5]$*. if we apply the new method on this interval with* $h = 0.1$ *along the linear interpolation, and with* $h = 0.125$ *along the quadratic interpolation and with* $h = 0.17$ *along the cubic interpolation, we obtain* $x = 1$ *which is the exact solution. But let we choose the initial interval* $[0.5, 2]$*. In Tables 1, 2 and 3, we present the results that obtained from executing the new method. In Table 1, we consider the linear interpolation with different mesh sizes on* $[0.5, 2]$*, Tables 2 and 3, respectively, contain the obtained results from applying the quadratic and cubic interpolations with different mesh sizes.*

As one can see, for a fixed h by increasing the degree of interpolation polynomials, we have better results, i.e., we have higher accuracy which confirms Theorem 1. For a fixed degree of interpolation polynomial, by decreasing h almost we have more accuracy. But in the case of the cubic interpolation, this is not true in some cases. This shows that in applying interpolation polynomials with higher degree, a medium value of h gives a good accuracy. Also, in each case, by decreasing h the computational cost increases and for a fixed h by increasing the degree of interpolation polynomials, the computational cost increases.

Example 2 *Consider the following function on* $[1, 3]$

$$f(x) = \sin x + x^2 \cos(e^x).$$

Figure 5 shows that this function has five roots on $[1, 3]$*. Now, we apply the new approach to estimate them. In Table 4, we consider the linear interpolation with different mesh sizes on* $[1, 3]$*. Also Table 5 contains results obtained from applying the quadratic interpolation with different mesh sizes.*

As one can see, for a fixed h almost we have more accuracy when the degree of the interpolating polynomial increases.

Example 3 *Consider the following function on* $[-1, 1]$

$$f(x) = \tan x + x^3 e^{\sin x + \cos x}.$$

By Figure 5, we find that this function has a root on $[-0.5, 0.5]$*. If we apply the new method on this interval, with* $h = 0.1$ *and linear interpolation we obtain the exact root* $x = 0$*. Even with* $h = 1$*, i.e., without dividing the interval* $[-0.5, 0.5]$*, and applying the quadratic and cubic interpolations we obtain the exact root.*

Example 4 *Consider the following function on interval* $[0, 10]$

$$f(x) = \frac{\sin x}{x}.$$

By Figure 5, we see that this function has three roots on $[0, 10]$*. The results from applying the new approach with different mesh sizes and degrees of polynomial are shown in Tables 6 and 7.*

From the reported numbers in Tables 6 and 7, we see that with smaller mesh sizes and polynomials of upper degree, almost we have more accuracy.

Example 5 *Consider the following function on* $[-1, 1]$

$$f(x) = \sinh x - x^2 \tan x.$$

Figure 5 shows that this function has three roots on $[-1, 1]$*. We report the obtained results using the new approach with different mesh sizes and degrees of interpolation polynomial, in Tables 8 and 9.*

Example 6 *Consider the following function on* $[-1, 1]$

$$f(x) = x^8 - xe^x.$$

Figure 5 shows that this function has one root on this interval. In fact, as can be seen this root is $x^* = 0$*, but let us find it by the new method. If we choose the interval* $[-0.5, 0.5]$ *and apply the new method with* $h = 0.1$ *and the linear interpolation polynomial we obtain the exact root* $x = 0$*. Also, by applying the quadratic and the cubic interpolation polynomials with* $h = 2$*, i.e., without dividing the interval* $[-0.5, 0.5]$*, we obtain the exact root* $x = 0$*.*

Example 7 *Consider the function*

$$f(x) = e^x - 1.5 - \tan^{-1} x.$$

By Figure 5, this function has one root on $[-15, -13]$*. The results obtained from the new approach with different mesh sizes and degrees of polynomial are shown in Tables 10 and 11.*

Example 8 *Consider*

$$f(x) = \cos x - xe^x + x^2.$$

By Figure 5, this function has one root on $[0, 1]$*. We use the new approach to estimate the root of* $f(x)$ *on this interval. The obtained results are shown in Tables 12 and 13.*

Example 9 *Consider*

$$f(x) = e^x - 4x^2.$$

By Figure 5, we see that this function has one root on $[0, 1]$*. The results obtained by executing the new approach are shown in 14 and 15.*

Table 1. Applying piecewise linear interpolation with different mesh sizes for Example 1

h	$root$	$\lvert f(root)\rvert$	Time (Sec.)
1.5×10^{-1}	1.002481663211218	0.002478588970175	0.0028
1.5×10^{-2}	1.000024979425922	$2.497911394083653\times10^{-5}$	0.0203
1.5×10^{-3}	1.000000249979193	$2.499791614343130\times10^{-7}$	0.2009
1.5×10^{-4}	1.000000002499979	$2.499979109488512\times10^{-9}$	1.8099
1.5×10^{-5}	1.000000000025000	$2.500000206819678\times10^{-11}$	17.6776

Table 2. Applying piecewise quadratic interpolation with different mesh sizes for Example 1

h	$root$	$\lvert f(root)\rvert$	Time (Sec.)
1.5×10^{-1}	1.000039493601601	$3.949282174951934\times10^{-5}$	0.0077
1.5×10^{-2}	1.000000041434118	$4.143411727811439\times10^{-8}$	0.0442
1.5×10^{-3}	1.000000000041826	$4.182609813964359\times10^{-11}$	0.3737
1.5×10^{-4}	1.000000000001205	$1.204591981717569\times10^{-12}$	3.5617
1.5×10^{-5}	1	0	35.3830

Table 3. Applying piecewise cubic interpolation with different mesh sizes for Example 1

h	$root$	$\lvert f(root)\rvert$	Time (Sec.)
1.5×10^{-1}	0.999999999999695	$3.048672425621144\times10^{-13}$	0.0207
1.5×10^{-2}	0.999999999923443	$7.655698297819263\times10^{-11}$	0.0619
1.5×10^{-3}	1.000000000232823	$2.328228720655940\times10^{-10}$	0.5598
1.5×10^{-4}	1.000000000000000	$2.220446049250313\times10^{-16}$	5.3436
1.5×10^{-5}	1	0	55.3164

Table 4. Applying piecewise linear interpolation with different mesh sizes for Example 2

h	$roots$	$\lvert f(root)\rvert$	Time (Sec.)
	1.423265561631496	0.089857348855157	
	2.085090207458152	0.043855873043206	
2×10^{-1}	2.362784447734398	1.342853250581783	0.0060
	2.692741833971670	3.866860193487434	
	2.927695131777520	8.667198356023061	
	1.437936164272049	0.000818816611662	
	2.086273194368227	0.001679667892384	
2×10^{-2}	2.386296798497964	0.009868611856714	0.0224
	2.653422275687298	0.008924654621447	
	2.847288686153410	0.016193447821624	
	1.438063739455481	$0.002751212680230\times10^{-3}$	
	2.086319938368043	$0.011813252343118\times10^{-3}$	
2×10^{-3}	2.386459407396437	$0.085748864212531\times10^{-3}$	0.2133
	2.653510025316480	$0.060326571126479\times10^{-3}$	
	2.847404277568455	$0.181016242704057\times10^{-3}$	
	1.438064139026840	$0.019411917939749\times10^{-5}$	
	2.086320263385221	$0.021609493650043\times10^{-5}$	
2×10^{-4}	2.3864608150275792	$0.103479095447945\times10^{-5}$	1.9957
	2.653510614808365	$0.077295509365305\times10^{-5}$	
	2.847405582422813	$0.022752305389728\times10^{-5}$	
	1.438064169130111	$0.014702502548758\times10^{-7}$	
	2.086320269438054	$0.001196586163488\times10^{-7}$	
2×10^{-5}	2.386460832189466	$0.019510805193690\times10^{-7}$	20.2828
	2.653510622382446	$0.077806149856308\times10^{-7}$	
	2.847405583942465	$0.169734550858891\times10^{-7}$	

Table 5. Applying piecewise quadratic interpolation with different mesh sizes for Example 2

h	*roots*	$\vert f(root)\vert$	Time (Sec.)
2×10^{-1}	1.437797181022304	0.001707031316506	0.0050
	2.084546074022266	0.063233186629051	
	2.387476619799538	0.061257401182087	
	2.640463283820497	1.297538518530415	
	2.830725342456636	2.222367388420808	
2×10^{-2}	1.438063732633222	0.000002794872295	0.0444
	2.086316651178341	0.000129105420773	
	2.386455046541641	0.000348190799036	
	2.653520678318183	0.001015904881206	
	2.847393530777692	0.001669966232352	
2×10^{-3}	1.438064169514777	$0.000099145913701\times10^{-5}$	0.4158
	2.086320265358900	$0.014567072026406\times10^{-5}$	
	2.386460825259738	$0.041899726099803\times10^{-5}$	
	2.653510635012532	$0.126817871154428\times10^{-5}$	
	2.847405600550434	$0.228407980967793\times10^{-5}$	
2×10^{-4}	1.438064169365771	$0.003787614666351\times10^{-8}$	3.9670
	2.086320269437931	$0.012404943738886\times10^{-8}$	
	2.386460832218397	$0.020991919313929\times10^{-8}$	
	2.653510622463481	$0.040595327099879\times10^{-8}$	
	2.847405584041350	$0.327283405932022\times10^{-8}$	
2×10^{-5}	1.438064169358210	$0.000105111475079\times10^{-7}$	39.7859
	2.086320269437904	$0.001250015646548\times10^{-7}$	
	2.386460832422985	$0.121026042521066\times10^{-7}$	
	2.653510622466619	$0.007230017140536\times10^{-7}$	
	2.847405584290039	$0.311833277533147\times10^{-7}$	

Table 6. Applying piecewise linear interpolation with different mesh sizes for Example 4

h	*roots*	$\vert f(root)\vert$	Time (Sec.)
10^{-1}	3.142372773174194	$0.248258103536299\times10^{-3}$	0.0026
	6.283392604992629	$0.032991382934222\times10^{-3}$	
	9.424991481436488	$0.022654732994280\times10^{-3}$	
10^{-2}	3.141596930997303	$0.136153924382458\times10^{-5}$	0.0197
	6.283188775076487	$0.055193262914768\times10^{-5}$	
	9.424780609967490	$0.028108857064407\times10^{-5}$	
10^{-3}	3.141592730427218	$0.244581114683089\times10^{-7}$	0.1993
	6.283185331222795	$0.038265955385632\times10^{-7}$	
	9.424777979081428	$0.019429686767671\times10^{-7}$	
10^{-4}	3.141592653806448	$0.689632920607629\times10^{-10}$	1.8493
	6.283185307379057	$0.317467029097984\times10^{-10}$	
	9.424777960951669	$0.193415030274949\times10^{-10}$	
10^{-5}	3.141592653595999	$0.197529751244538\times10^{-11}$	18.4089
	6.283185307183549	$0.063070035111195\times10^{-11}$	
	9.424777960771102	$0.018278401330503\times10^{-11}$	

5. Concluding remark

The problem of finding roots of a nonlinear function appears in many fields of science and engineering. In this paper, we presented and analyzed a new approach based on partitioning an interval into some subintervals to estimate the roots of nonlinear functions. The introduced method can be used to find the roots of a more class of nonlinear functions without computing their derivatives. Also it can be generalized to estimate the roots of the multivariable functions in higher dimensions. The new approach is capable to find all the roots of a function on the determined interval $[a, b]$. Numerical experiments showed the effectiveness of the new approach.

Table 7. Applying piecewise quadratic interpolation with different mesh sizes for Example 4

h	$roots$	$\lvert f(root)\rvert$	Time (Sec.)
10^{-1}	3.141591329586333	$0.042144356837710\times10^{-5}$	0.0048
	6.283191968799295	$0.106022858158360\times10^{-5}$	
	9.424770714926494	$0.076880839918186\times10^{-5}$	
10^{-2}	3.141592650618887	$0.945668991712094\times10^{-9}$	0.0420
	6.283185301618809	$0.885025284810616\times10^{-9}$	
	9.424777959907821	$0.091414175184169\times10^{-9}$	
10^{-3}	3.141592653590999	$0.038388930783063\times10^{-11}$	0.3839
	6.283185307199156	$0.311464287833273\times10^{-11}$	
	9.424777960819229	$0.528919989386322\times10^{-11}$	
10^{-4}	3.141592653599132	$0.297271946054740\times10^{-11}$	3.6800
	6.283185307208281	$0.456681346688490\times10^{-11}$	
	9.424777960720419	$0.519485072281183\times10^{-11}$	
10^{-5}	3.141592653309660	$0.089169221891266\times10^{-9}$	38.1534
	6.283185308195679	$0.161716182598050\times10^{-9}$	
	9.424777957057623	$0.393829607446302\times10^{-9}$	

Table 8. Applying piecewise linear interpolation with different mesh sizes for Example 5

h	$roots$	$\lvert f(root)\rvert$	Time (Sec.)
2×10^{-1}	-0.874961843703970	0.074328028941983	0.0041
	0	0	
	0.874961843703970	0.074328028941983	
2×10^{-2}	-0.901868808981154	$0.282014567155153\times10^{-3}$	0.0205
	0	0	
	0.901868808981154	$0.282014567155153\times10^{-3}$	
2×10^{-3}	-0.901963811046113	$0.575347403941606\times10^{-6}$	0.2110
	0	0	
	0.901963811046113	$0.575347403941606\times10^{-6}$	
2×10^{-4}	-0.901963989030390	$0.479396948804833\times10^{-7}$	1.9420
	0	0	
	0.901963989030389	$0.479396951025279\times10^{-7}$	
2×10^{-5}	-0.901964005033055	$0.520149034954898\times10^{-9}$	20.2546
	0	0	
	0.901964005033055	$0.520148812910293\times10^{-9}$	

Table 9. Applying piecewise quadratic interpolation with different mesh sizes for Example 5

h	$roots$	$\lvert f(root)\rvert$	Time (Sec.)
2×10^{-1}	-0.901884158789132	$0.236551396225027\times10^{-3}$	0.0043
	0	0	
	0.901884158789130	$0.236551396230356\times10^{-3}$	
2×10^{-2}	-0.901965239910942	$0.365871711127674\times10^{-5}$	0.0421
	0	0	
	0.901965239910932	$0.365871707974641\times10^{-5}$	
2×10^{-3}	-0.901964004925138	$0.839932567942014\times10^{-9}$	0.4002
	0	0	
	0.901964004924705	$0.841215319624666\times10^{-9}$	
2×10^{-4}	-0.901964005207871	$0.212940776123105\times10^{-11}$	3.8387
	0	0	
	0.901964005207788	$0.237387887125351\times10^{-11}$	
2×10^{-5}	-0.901964005221988	$0.397042398958547\times10^{-10}$	40.7914
	0	0	
	0.901964005195030	$0.401794153503943\times10^{-10}$	

Table 10. Applying piecewise linear interpolation with different mesh sizes for Example 7

h	$root$	$\|f(root)\|$	Time (Sec.)
2×10^{-1}	-14.101974620425656	$3.526252680741138\times10^{-6}$	0.0038
2×10^{-2}	-14.101271449371303	$8.388364003408810\times10^{-9}$	0.0209
2×10^{-3}	-14.101269838105679	$3.270315129810797\times10^{-10}$	0.2110

Table 11. Applying piecewise quadratic interpolation with different mesh sizes for Example 7

h	$root$	$\|f(root)\|$	Time (Sec.)
2×10^{-1}	-14.101269709928433	$-3.142528459676441\times10^{-10}$	0.0087
2×10^{-2}	-14.101269773764367	$5.125011526274648\times10^{-12}$	0.0443
2×10^{-3}	-14.101269772742056	$1.043609643147647\times10^{-14}$	0.3964

Table 12. Applying piecewise linear interpolation with different mesh sizes for Example 8

h	$root$	$\|f(root)\|$	Time (Sec.)
10^{-1}	0.637295803823317	0.004498320539004	0.0030
10^{-2}	0.639148010047697	$1.475432710396074\times10^{-5}$	0.0201
10^{-3}	0.639153994091137	$2.478527552085552\times10^{-7}$	0.2178

Table 13. Applying piecewise quadratic interpolation with different mesh sizes for Example 8

h	$root$	$\|f(root)\|$	Time (Sec.)
10^{-1}	0.639165394620634	$2.738960439940819\times10^{-5}$	0.0069
10^{-2}	0.639154082455807	$3.363874623296681\times10^{-8}$	0.0432
10^{-3}	0.639154096351640	$4.759215244121151\times10^{-11}$	0.3996

Table 14. Applying piecewise linear interpolation with different mesh sizes for Example 9

h	$root$	$\|f(root)\|$	Time (Sec.)
10^{-1}	0.713846233005084	0.003523749696264	0.0046
10^{-2}	0.714785689013024	$7.431266240054413\times10^{-5}$	0.0185
10^{-3}	0.714805785523238	$4.660917953813737\times10^{-7}$	0.2891

Table 15. Applying piecewise quadratic interpolation with different mesh sizes for Example 9

h	$root$	$\|f(root)\|$	Time (Sec.)
10^{-1}	0.714801687027082	$1.552660761339197\times10^{-5}$	0.0066
10^{-2}	0.714805911913587	$1.650623637772242\times10^{-9}$	0.0435
10^{-3}	0.714805912367387	$1.693667428526169\times10^{-11}$	0.3792

Acknowledgment

The author would like to thank the Shahid Chamran University of Ahvaz for financial support under the grant number SCU.MM1400.33518.

References

1. G. Alefeld and F. A. Potra. Some efficient methods for enclosing simple zeroes of nonlinear equations. *BIT*, 32:334–344, 1992.
2. S. Amat, S. Busquier, and J. M. Gutirrez. Geometric constructions of iterative functions to solve nonlinear equations. *Journal of Computational and Applied Mathematics*, 157:197–205, 2003.
3. K. E. Atkinson. *An Introduction to Numerical Analysis*. NY: John Wiley and Sons, New York, 1993.
4. J. Biazar and B. Ghanbari. A new third-order family of nonlinear solvers for multiple roots. *Computational and Applied Mathematics*, 59:3315–3319, 2010.
5. M. Bisheh-Niasar and K. Gdawiec. Bisheh-niasarsaadatmandi root finding method via the s-iteration with periodic parameters and its polynomiography. *Mathematics and Computers in Simulation*, 160:1–12, 2019.
6. D. Braess. *Finite Elements*. Cambridge University Press, 2007.
7. S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Method, Third Edition*. Springer, 2008.
8. Z. Chen. *Finite Element Methods and Their Applications*. Springer Verlag, Berlin Heidelberg, 2005.
9. C. Chun. Construction of newton-like iteration methods for solving nonlinear equations. *Numerische Mathematik*, 104:297–315, 2006.
10. C. Chun. A new iterative method for solving nonlinear equations. *Applied Mathematics and Computation*, 178:415–422, 2006.
11. P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. SIAM, Paris, 2002.
12. M. Dehghan and M. Hajarian. Some derivative free quadratic and cubic convergence iterative formulas for solving nonlinear equations. *Computational and Applied Mathematics*, 29:19–30, 2010.
13. M. Dehghan and M. Hajarian. On some cubic convergence iterative formulae without derivatives for solving nonlinear equations. *Communications in Numerical Methods in Engineering*, 27:722–731, 2011.
14. M. Dowell and D. Jarratt. A modified regula falsi method for computing the root of an equation. *BIT*, 11:168–174, 1971.
15. M. Dowell and D. Jarratt. The pegasus method for computing the root of an equation. *BIT*, 12:503–508, 1972.
16. R. T. Fenner. *Finite Element Methods for Engineers*. Imperial College Press, London, 1996.
17. K. Gdawiec, W. Kotarski, and A. Lisowska. On the robust newtons method with the mann iteration and the artistic patterns from its dynamics. *Nonlinear Dynamics*, 104:297–331, 2021.
18. M. S. Gockenbach. *Understanding and Implementing the Finite Element Method*. SIAM, 2006.
19. M. Iwata, A. Miyawaki-Kuwakado, E. Yoshida, S. Komori, and F. Shiraishi. Evaluation of an s-system root-finding method for estimating parameters in a metabolic reaction model. *Mathematical Biosciences*, 301:21–31, 2018.
20. C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, 1987.
21. K. Kalorkoti. Tree breadth of the continued fractions root finding method. In *New Trends in Algebras and Combinatorics*, pages 203–227. Proceedings of the 3rd International Congress in Algebras and Combinatorics, 2020.
22. Y. I. Kim and Y. H. Geum. A cubic-order variant of newton's method for finding multiple roots of nonlinear equations. *Computers & Mathematics with Applications*, 62:1634–1640, 2011.
23. R. F. King. A family of fourth order methods for nonlinear equations. *SIAM Journal on Numerical Analysis*, 10:876–879, 1973.
24. J. Kou, X. Wang, and Y. Li. Some eighth-order root-finding three-step methods. *Communications in Nonlinear Science and Numerical Simulation*, 15:536–544, 2010.
25. J. M. McNamee. Numerical methods for roots of polynomials. In *Part 1, Studies in Computational Mathematics*. Elsevier Science, Cambridge, MA, 2007.
26. D. E. Muller. A method for solving algebraic equations using an automatic computer. *Mathematical Tables and Other Aids to Computation*, 10:208–215, 1956.
27. A. M. Ostrowski. *Solutions of Equations and System of Equations*. New York: Academic Press, 1960.
28. V. Y. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Review*, 39:187–220, 1997.
29. M. S. Petkovic, J. Dunic, and M. Miloevc. Traub's accelerating generator of iterative root finding methods. *Applied Mathematics Letters*, 24:1443–1448, 2011.
30. A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, Berlin Heidelberg, 2007.
31. S. Qureshi, H. Ramos, and A. K. Soomro. A new nonlinear ninth-order root-finding method with error analysis and basins of attraction. *Mathematics*, 9, 2021, doi: 10.3390/math9161996.
32. F. K. Richard. An improved pegasus method for root finding. *BIT*, 13:423–427, 1973.
33. W. Rudin. *Real and Complex Analysis*. Tata McGraw-Hill, New Delhi, 1983.
34. O. Ste and J. Zur. A newton method for harmonic mappings in the plane. *IMA Journal of Numerical Analysis*, 40:2777–2801, 2020.
35. A. O. Umar, M. Mamat, and M. Y. Waziri. Solving dual fuzzy nonlinear equations via modified stirling's method. *Journal of Information System and Technology Management*, 4(14):84–91, 2019.
36. X. Wu and D. Fu. New high order convergence iteration methods without employing derivatives for solving nonlinear equations. *Computers & Mathematics with Applications*, 41:489–495, 2001.