Appl. Math. J. Chinese Univ.
2025, 40(4): 833-852

Progressive explicit formulae for root-finding problems based on reparameterization

WANG Hui QIAN Cheng CHEN Xiao-diao*

Abstract. This paper presents an explicit formula based on reparameterization technique for progressively computing a simple root of a smooth function, which may have wide applications in robotics, geomagnetic navigation, geometric processing and computer graphics. Comparing with Newton-like method, it can achieve convergence rate 2 by adding one more functional evaluation, improve the computational stability and ensure the convergence, and also obtain higher convergence rate and higher efficiency index. Compared with clipping methods for polynomials, it doesn't need to bound the polynomials, directly bound the roots and can also work well for non-polynomial functions with much higher computational efficiency. Comparing with previous progressive methods, it achieves a much higher computational efficiency and is extended to solve bivariate equation system. Numerical examples show its much better performance on approximation error, computational efficiency and computational stability.

§1 Introduction

The root-finding problem is a common and important problem in science and engineering and it has wide applications in robotics [8], geomagnetic navigation, computer aided geometric design [17] and computer graphics [1, 3].

There are so many references on the root-finding problems. For the root-finding problem of a univariate equation, there are Newton-like methods [9, 15, 22, 23, 25, 26, 27, 28, 30, 31, 32, 33], clipping methods based on Bernstein-Bézier form [2, 4, 5, 13, 14, 17, 18, 21, 29], interval methods [19, 20] and progressive methods [6, 7]. The works on solving a system of non-linear equations are referred to [1, 3, 12, 24] and the references therein.

The progressive method in [7] provides explicit derivative-free formula for the root-finding problem of a univariate equation, which achieves the optimal efficiency index in the conjecture

Received: 2022-08-01. Revised: 2023-01-06.
MR Subject Classification: 65D17, 68W25.
Keywords: root-finding, re-parameterization-based method, clipping method, numerical iterative method, convergence order, non-linear equation system, progressive explicit formulae.
Digital Object Identifier(DOI): https://doi.org/10.1007/s11766-025-4808-6.
Supported by the National Natural Science Foundation of China (61972120).
*Corresponding author.

[15] and obtains much better robustness than Newton-like methods. At the moment, it seems to be difficult to extend the progressive method in [7] to the bivariate equation system case, partially because that the root-finding problem of two approximation bivariate equations itself is not easy to be solved. Moreover, for a multiple root case, it is necessary to speed up the convergence rate by using the derivatives, which is not considered in [7].

This paper presents a reparameterization-based method (RBM) to construct explicit formulae of the root-finding problems of both univariate cases and bivariate cases. It achieves the same robustness as that of the progressive method in [7]; at the same time, it also provides the corresponding explicit formula by using the derivatives. Numerical examples show that the RBM achieves much better computational efficiency than that of the progressive method in [7].

§2 The RBM method for univariate cases

2.1 Preliminary theory

For the sake of convenience, we also introduce Theorem 3.5.1 in Page 67, Chapter 3.5 of [10] as follows.

Theorem 1. Let $w_0, w_1, \cdots, w_r$ be $r+1$ distinct points in $[a,b]$, and $n_0, \cdots, n_r$ be $r+1$ integers ≥ 1. Let $N = n_0 + n_1 + \cdots + n_r$. Suppose that $g(t)$ is a polynomial of degree N such that $g^{(i)}(w_j) = f^{(i)}(w_j), \quad i = 0, \cdots, n_j - 1, \quad j = 0, \cdots, r$. There exists $\xi_0(t) \in [a,b]$ such that

$$f(t) - g(t) = \frac{f^{(N)}(\xi_0(t))}{N!} \prod_{i=0}^{r} (t - w_i)^{n_i}.$$

From Theorem 1, if a function $g(s)$ interpolates another function $f(t)$ at several points such that $f(t_j) = g(s_j), \quad t_j, s_j \in [a,b], \quad j = 0, 1, \cdots, r$, and suppose that $\phi(s)$ is a monotonous function such that $t_j = \phi(s_j), \ j = 0, 1, \cdots, r, \ \forall s \in [a,b]$, there exists $\xi_1(s) \in [a,b]$ such that

$$f(t_j) = f(\phi(s_j)) = \bar{f}(s_j) = g(s_j), \quad j = 0, 1, \cdots, r,$$
$$f(\phi(s)) - g(s) = \frac{(\bar{f} - g)^{(r+1)}(\xi_1(s))}{(r+1)!} \prod_{j=0}^{r} (s - s_j),$$

which means that the error $f(\phi(s)) - g(s)$ can be of $O((b-a)^{r+1})$.

Furthermore, by selecting a suitable monotonous function $\hat{\phi}(s)$ such that

$$\hat{\phi}(s_j) = t_j \quad \text{and} \quad \hat{\phi}'(s_j) = \frac{g'(s_j)}{f'(t_j)}, \quad j = 0, 1, \cdots, r,$$

it can be verified that $\hat{f}(s_j) = g(s_j), \quad \hat{f}'(s_j) = g'(s_j), \quad j = 0, 1, \cdots, r$, where $\hat{f}(s) = f(\hat{\phi}(s))$. From Theorem 1, there exists $\xi_2(s) \in [a,b]$ such that

$$\hat{f}(s) - g(s) = \frac{(\hat{f} - g)^{(2r+2)}(\xi_2(s))}{(2r+2)!} \prod_{j=0}^{r} (s - s_j)^2,$$

which means that the error $f(\hat{\phi}(s)) - g(s)$ can be of $O((b-a)^{2r+2})$. Both $\phi(s)$ and $\hat{\phi}(s)$ are called reparameterization functions. Fig. 1 shows an example of $\sin(t)$. In Fig. 1(b), the errors $\sin(t) - p(t)$ and $\sin(t) - \bar{p}(t)$ are plotted in solid black and in dashed red, respectively, where $\bar{p}(t) = p(\phi(t))$, $p(t)$ is the quadratic polynomial interpolating $\sin(t)$ at three points $t_1 = 0, t_2 = \pi/4, t_3 = \pi/2$, and $\phi(t)$ is a quintic polynomial satisfying $\phi(t_i) = t_i$ and $\bar{p}'(t_i) =$

$\sin'(t_i)$, $i = 1, 2, 3$. In principle, $p(t)$ interpolates position of $\sin(t)$ at three points and achieves approximation order 3, while while $\bar{p}(t) = p(\phi(t))$ interpolates both position and derivative of $\sin(t)$ at three points and achieves approximation order 6, which means that $p(\phi(t))$ is expected to have much better approximation error than that of $p(t)$. As shown in Fig. 1(b), the approximation error can be improved by using reparameterization.

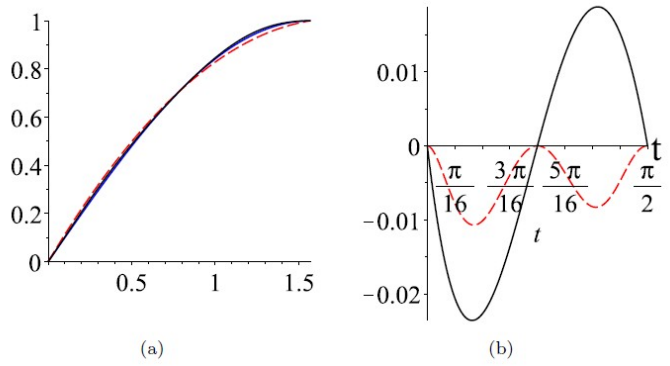


Figure 1. Improve the approximation error by using reparameterization: (a) The plots of $\sin(t)$ in solid blue, $p(t)$ in dashed red and $p(\phi(t))$ in solid black; and (b)the curves in solid black and in dashed red are the errors $\sin(t) - p(t)$ and $\sin(t) - p(\phi(t))$, respectively.

Let $s^\star$ be the root satisfying $g(s^\star) = 0$, one has that $f(\phi(s^\star)) = f(\phi(s^\star)) - g(s^\star) \approx 0$, which means that $\phi(s^\star)$ can be used to approximate the root $t^\star$ satisfying $f(t^\star) = 0$. So it is the same with $\hat{\phi}(s^\star)$.

Suppose that κ_1 and κ_2 are two real numbers within $[a, b]$. We have the following lemma for bounding the root $t^\star$ of $f(t)$.

Lemma 1. If $|\kappa_2 - t^\star| < \dfrac{\theta}{1+\theta} \cdot |\kappa_1 - t^\star|$, where $\theta > 0$, we have $\kappa_2 + \theta(\kappa_2 - \kappa_1)$ and κ_1 bound $t^\star$.

Proof. Without loss generality, assume that $\kappa_1 < \kappa_2$.

(1) Firstly, we claim that $\kappa_1 < t^\star$. Otherwise, suppose that $\kappa_1 \geq t^\star$, one has that $|\kappa_2 - t^\star| \geq |\kappa_1 - t^\star|$, which conflicts with the assumption that $|\kappa_2 - t^\star| < \dfrac{\theta}{1+\theta} \cdot |\kappa_1 - t^\star| \leq |\kappa_1 - t^\star|$.

(2) Secondly, if $\kappa_2 \geq t^\star$, one has $\kappa_2 + \theta(\kappa_2 - \kappa_1) \geq \kappa_2 \geq t^\star$, combining with $\kappa_1 < t^\star$, we have $\kappa_2 + \theta(\kappa_2 - \kappa_1)$ and κ_1 bound $t^\star$.

(3) Finally, if $\kappa_2 < t^\star$, we will prove that $\kappa_2 + \theta(\kappa_2 - \kappa_1)$ and κ_1 bound $t^\star$. Combining with the assumption $|\kappa_2 - t^\star| < \dfrac{\theta}{1+\theta} \cdot |\kappa_1 - t^\star|$, we have

$$0 > (1+\theta) \cdot (t^\star - \kappa_2) - \theta \cdot (t^\star - \kappa_1) = t^\star - (\kappa_2 + \theta \cdot (\kappa_2 - \kappa_1)).$$

Combining with $\kappa_1 < t^\star$, we have

$$\kappa_1 < t^\star < \kappa_2 + \theta \cdot (\kappa_2 - \kappa_1),$$

which means $\kappa_2 + \theta(\kappa_2 - \kappa_1)$ and κ_1 bound $t^\star$.

From the above discussions, the proof has been completed.

2.2 The idea and the algorithm of RBM for a derivative-free case

Given a smooth function $f(t)$ and the interval $[a,b]$ as well. For the case that $f(t)$ is a polynomial, one can isolate the multiple roots by using the clipping methods such as the ones in [5, 17]. While for a case that $f(t)$ is a non-polynomial function, this section also provides an alternative way for isolating the roots as follows. Let r_i be the root of $F_i(t)$, $i=2,3$, where $F_i(t)$ is a polynomial of degree i and interpolates $f(t)$ at $i+1$ points (or derivatives as well), and note that the condition $2|r_3-t^\star|<|r_2-t^\star|$ can be satisfied, from Lemma 1, one has that $2r_3-r_2$ and r_2 can bound $t^\star$. The root isolation can be done in the above way.

After the above root isolation process, this paper assumes that $f(t)$ has a unique root $t^\star$ within $[a,b]$.

In [7], $f(t)$ is approximated by a rational polynomial $P(t)=(\alpha_{i,1}t+\alpha_{i,0})/(1+\sum\limits_{j=1}^{i}\alpha_{i,j+1}t^j)$, where $\alpha_{i,j}$ are $i+2$ real numbers determined by the $i+2$ constraints $f(t_j)=P(t_j)$, $j=0,1,\cdots,i+1$. Then the root $t=-\alpha_{i,0}/\alpha_{i,1}$ is used to progressively approximate the root $t^\star$ of $f(t)$, $i=1,2,\cdots$.

In this paper, the RBM method utilizes $\mathbf{A}_i(s)=\frac{(s,L(s))}{g_i(s)}$ instead to interpolate $\mathbf{C}(t)=(t,f(t))$ at several points $t=t_j$, $j=0,1,\cdots,i$, where $L(s)=\alpha+\beta s$ is a linear function satisfying

$$\alpha=\frac{f(a)b-f(b)a}{b-a},\quad \beta=\frac{f(b)-f(a)}{b-a},\quad L(a)=f(a)\ \text{ and }\ L(b)=f(b), \tag{1}$$

which has a root $s^\star\in[a,b]$ such that $L(s^\star)=0$.

Similarly as the one in [7], all of the points t_j are nearby or close to the root $t^\star$, $j=0,1,\cdots,i$. For the case that $f(t)$ is a polynomial, the more interpolation points $\{t_j\}_{j=0}^{i}$, the smaller approximation error $|L(s)/g_i(s)-f(\phi_i(s))|$, especially for the place nearby $t^\star$. It means that $f(\phi_i(s^\star))\approx L(s^\star)/g_i(s^\star)=0$, and $\phi_i(s^\star)$ can be used to approximate the root $t^\star$ of $f(t)$.

The RBM provides an explicit formula for directly computing the value of s_i for any given $t_i\in(a,b)$, and it does not need to compute the expression of $g_i(s)$ at all, which achieves much better computational efficiency than those of the progressive methods in [6, 7]. With the values of s_i, both the polynomial $g_i(s)$ and the reparameterization function $\phi_i(s)=\frac{s}{g_i(s)}$ can be determined by using the corresponding explicit formulas. The details are as follows.

From Theorem 1, both $\phi_i(s)=\frac{s}{g_i(s)}$ and $G_i(s)=\frac{L(s)}{g_i(s)}$ can be used to approximate t and $f(t)$, respectively, i.e.,

$$t\approx\phi_i(s)\ \text{ and }\ f(t)\approx G_i(s)\ \text{ and }\ f(\phi_i(s))\approx G_i(s). \tag{2}$$

From Eq. (2), one has $f(\phi_i(s^\star))\approx G_i(s^\star)=0$, which leads to

$$\phi_i(s^\star)\approx t^\star. \tag{3}$$

Given t_j, $j=0,1,\cdots,i$, we determine the values of $g_i(s^\star)$ and $t_{i+1}=\phi_i(s^\star)=\frac{s^\star}{g_i(s^\star)}$ as follows. Firstly, from

$$\mathbf{C}(t_j) = \mathbf{A}_i(s_j), \quad j = 0, 1, \cdots, i, \tag{4}$$

one has $\mathbf{C}(t_j) \times \mathbf{A}_i(s_j) = 0$, which leads to

$$t_j L(s_j) - f(t_j) s_j = 0, \quad j = 0, 1, \cdots, i. \tag{5}$$

By solving the linear Eq. (5), we obtain the value of s_j.

Secondly, by using $t_j = \phi_i(s_j) = \frac{s_j}{g_i(s_j)}$, one has

$$g_i(s_j) = \frac{s_j}{t_j} = \gamma_j, \quad j = 0, 1, \cdots, i. \tag{6}$$

From Eq. (6), the unique polynomial $g_i(s)$ of degree i can be explicitly expressed by the following progressive formula

$$g_1(s) = 1, \quad g_i(s) = g_{i-1}(s) + \frac{\frac{s_i}{t_i} - g_{i-1}(s_i)}{\prod\limits_{j=0}^{i-1}(s_i - s_j)} \prod_{j=0}^{i-1}(s - s_j), \quad i \geq 2, \tag{7}$$

and then, $t_{i+1} = \frac{s^\star}{g_i(s^\star)}$ is obtained.

The outline of the algorithm is described as follows.

Algorithm 1. The RBM method for solving the unique root $t^\star \in [a, b]$ of $f(t)$.

Input: The given smooth function $f(t)$, the interval $[a, b]$ and the tolerance ε.

Output: The approximation of $t^\star$.

(1) **Begin:** Let $i = 2$, $g_1(s) = s$, $t_0 = s_0 = a$, $t_1 = s_1 = b$ and $t_2 = \frac{a+b}{2}$;
Compute α, β and $s^\star = -\frac{\alpha}{\beta}$.

(2) If $|t_i - t_{i-1}| < \varepsilon$, go to Step (6); Otherwise, go to Step (3).

(3) By solving Eq. (2), one obtains the values of s_i and γ_i.

(4) By using Eq. (4), one obtains $t_{i+1} = \frac{s^\star}{g_i(s^\star)}$.

(5) By setting $i = i + 1$, go to Step (2).

(6) **End:** Output t_i as the approximation of $t^\star$.

2.3 The idea and the algorithm of RBM by using derivatives

In this section, it uses $\hat{\mathbf{A}}_i(s) = \frac{(s, L(s))}{\hat{g}_i(s)}$ instead which is tangent with $\mathbf{C}(t) = (t, f(t))$ at several points $\mathbf{C}(t_j)$, $j = 0, 1, \cdots, i$, where $L(s) = \alpha + \beta s$ is determined by Eq. (1). Similarly, one has

$$t \approx \hat{\phi}_i(s) = \frac{s}{\hat{g}_i(s)} \quad \text{and} \quad \hat{f}(s) = f(\hat{\phi}_i(s)) \approx \hat{G}_i(s) = \frac{L(s)}{\hat{g}_i(s)}. \tag{8}$$

From Eq. (8), one has $f(\hat{\phi}_i(s^\star)) \approx \hat{G}_i(s^\star) = 0$, which leads to

$$\hat{\phi}_i(s^\star) \approx t^\star. \tag{9}$$

Given t_j, $j = 0, 1, \cdots, i$, we determine the values of $g_i(s^\star)$ and $\hat{t}_{i+1} = \hat{\phi}_i(s^\star) = \frac{s^\star}{\hat{g}_i(s^\star)}$ as follows.

Firstly, from $\mathbf{C}(\hat{t}_j) = \hat{\mathbf{A}}_i(s_j)$, one obtains the linear Eq. (2) for determining the value of s_j, $j = 0, 1, \cdots, i$.

Secondly, by using $\hat{t}_j = \hat{\phi}_i(s_j) = \frac{s_j}{\hat{g}_i(s_j)}$, one has

$$\hat{g}_i(s_j) = \frac{s_j}{\hat{t}_j} = \gamma_j, \quad j = 0, 1, \cdots, i. \tag{10}$$

Thirdly, combining Eq. (7) with $\hat{f}'(s_j) = \hat{G}'_i(s_j), j = 0, 1, \cdots, i$, one has

$$\hat{f}'(s_j) = f'(\hat{t}_j) \cdot (\tfrac{1}{\gamma_j} - \tfrac{s_j \cdot \hat{g}'_i(s_j)}{\gamma_j^2}) = \hat{G}'_i(s_j) = \tfrac{L'(s_j)}{\gamma_j} - \tfrac{L(s_j) \cdot \hat{g}'_i(s_j)}{\gamma_j^2}, \tag{11}$$

which leads to

$$\hat{g}'_i(s_j) = \tfrac{(f'(\hat{t}_j) - L'(s_j)) \cdot \gamma_j}{f'(\hat{t}_j) s_j - L(s_j)} = \kappa_j, \quad j = 0, 1, \cdots, i. \tag{12}$$

Finally, let $h_i(s) = \prod_{j=0}^{i} (s - s_j)^2$, combining Eq. (10) with Eq. (12), the unique polynomial $\hat{g}_i(s)$ of degree $2i + 1$ can be explicitly expressed by the following progressive formula

$$\begin{aligned} &\hat{g}_0(s) = \kappa_0 s + (\gamma_0 - \kappa_0 s_0), \\ &\hat{g}_i(s) = \hat{g}_{i-1}(s) + (\lambda_{1,i} s + \lambda_{2,i}) h_{i-1}(s), \quad i \geq 1, \end{aligned} \tag{13}$$

where

$$\begin{cases} \lambda_{1,i} = \frac{-\hat{g}'_{i-1}(s_i) + \kappa_i}{h_{i-1}(s_i)} - \frac{h'_{i-1}(s_i)(\gamma_i - \hat{g}_{i-1}(s_i))}{h_{i-1}(s_i)^2}, \\ \lambda_{2,i} = \frac{\hat{g}'_{i-1}(s_i) s_i - \kappa_i s_i + \gamma_i - \hat{g}_{i-1}(s_i)}{h_{i-1}(s_i)} + \frac{h'_{i-1}(s_i) s_i (\gamma_i - \hat{g}_{i-1}(s_i))}{h_{i-1}(s_i)^2}. \end{cases}$$

And $\hat{t}_{i+1} = \frac{s^\star}{\hat{g}_i(s^\star)}$ is obtained.

The corresponding algorithm is similar to Algorithm 1, which uses $\hat{g}_i(s^\star)$ instead of $g_i(s^\star)$ for computing the value of $\hat{t}_{i+1}$.

2.4 Discussions on the convergence order

Let $D_i(s) = f(\phi_i(s)) - G_i(s)$. Combining Eq. (2), Eq. (4) with Eq. (6), one has

$$D_i(s_j) = f(\phi_i(s_j)) - G_i(s_j) = 0, \quad j = 0, 1, \cdots, i. \tag{14}$$

Theorem 2. For $\forall s \in [a, b]$, if $|D_i^{(i+1)}(s)| < M_1$, where M_1 is a positive real number and $i > 2$, we have

$$|D_i(s)| < \tfrac{M_1}{(i+1)!} \prod_{j=0}^{i} |(s - s_j)|. \tag{15}$$

Proof. Combining Eq. (14) with Theorem 1, there exists $\xi_3(s) \in [a, b]$ such that

$$D_i(s) = f(\phi_i(s)) - G_i(s) = \tfrac{D_i^{(i+1)}(\xi_3(s))}{(i+1)!} \prod_{j=0}^{i} (s - s_j). \tag{16}$$

Combining Eq. (16) with $|D_i^{(i+1)}(s)| < M_1$, we obtain Eq. (15), and the proof is completed.

Let $\hat{D}_i(s) = f(\hat{\phi}_i(s)) - \hat{G}_i(s)$. Combining Eq. (5), Eq. (7) with Eq. (8), one has

$$\hat{D}_i(s_j) = \hat{D}'_i(s_j) = 0, \quad j = 0, 1, \cdots, i. \tag{17}$$

Theorem 3. For $\forall s \in [a, b]$, if $|\hat{D}_i^{(2i+2)}(s)| < M_2$, where M_2 is a positive real number and $i > 2$, we have

$$|\hat{D}_i(s)| = |f(\hat{\phi}_i(s)) - \hat{G}_i(s)| < \tfrac{M_2}{(2i+2)!} \prod_{j=0}^{i} (s - s_j)^2. \tag{18}$$

Proof. Combining Eq. (17) with Theorem 1, there exists $\xi_4(s) \in [a, b]$ such that

$$\hat{D}_i(s) = \tfrac{\hat{D}_i^{(2i+2)}(\xi_4(s))}{(2i+2)!} \prod_{j=0}^{i} (s - s_j)^2. \tag{19}$$

Combining Eq. (19) with $|\hat{D}_i^{(2i+1)}(s)| < M_2$, we obtain Eq. (18), and the proof is completed.

From Theorems 2 and 3, we have the following corollaries, and similar method for their proofs can be referred to [11].

Corollary 1. Suppose that for $\forall s \in [a, b]$, when $f(t)$ is a polynomial such that $|D_i^{(i+1)}(s)| < M_1$, where M_1 is a constant number, $t^\star$ is a unique but simple root of $f(t)$ within [a,b] and $G(s^\star) = 0$, we have

$$|\phi_i(s^\star) - t^\star| = O(|\phi_{i-1}(s^\star) - t^\star|^3), \quad i \geq 3.$$

Proof. Note that $\phi_i(s)$ is a reparameterization function, combining Theorem 2, $t_j = \phi_i(s_j)$, $t_{i+1} = \phi_i(s^\star)$ with the assumption, one has

$$\begin{aligned} |t_{i+1} - t^\star| = \; & |\phi_i(s^\star) - t^\star| \\ & = O(|f(\phi_i(s^\star)) - f(t^\star)|) = O(|f(\phi_i(s^\star))|) \\ & = O(|f(\phi_i(s^\star)) - G(s^\star)|) = O(\prod_{j=0}^{i} |s^\star - s_j|) \\ & = O(\prod_{j=0}^{i} |\phi_i(s^\star) - \phi_i(s_j)|) = O(\prod_{j=0}^{i} |t_{i+1} - t_j|) \\ & = O(\prod_{j=0}^{i} |(t_{i+1} - t^\star) - (t_j - t^\star)|) = O(\prod_{j=0}^{i} |t_j - t^\star|) \\ & = O(\prod_{j=0}^{i-1} |t_j - t^\star|) \cdot |t_i - t^\star| = O(|t_i - t^\star|^2), \end{aligned} \tag{20}$$

and the proof is completed.

Corollary 2. Suppose that for $\forall s \in [a, b]$, when $f(t)$ is a polynomial such that $|\hat{D}_i^{(2i+2)}(s)| < M_2$, where M_2 is a constant number, $t^\star$ is a unique but simple root of $f(t)$ within [a,b] and $\hat{G}(s^\star) = 0$, we have

$$|\hat{\phi}_i(s^\star) - t^\star| = O(|\hat{\phi}_{i-1}(s^\star) - t^\star|^3), \quad i \geq 3.$$

Proof. Note that $\hat{\phi}_i(s)$ is a reparameterization function, combining Theorem 3, $\hat{t}_j = \hat{\phi}_i(s_j)$, $\hat{t}_{i+1} = \hat{\phi}_i(s^\star)$ with the assumption, one has

$$\begin{aligned} |\hat{t}_{i+1} - t^\star| = \; & |\hat{\phi}_i(s^\star) - t^\star| \\ & = O(|f(\hat{\phi}_i(s^\star)) - f(t^\star)|) = O(|f(\hat{\phi}_i(s^\star))|) \\ & = O(|f(\hat{\phi}_i(s^\star)) - \hat{G}(s^\star)|) = O(\prod_{j=0}^{i} |s^\star - s_j|^2) \\ & = O(\prod_{j=0}^{i} |\hat{\phi}_i(s^\star) - \hat{\phi}_i(s_j)|) = O(\prod_{j=0}^{i} |\hat{t}_{i+1} - \hat{t}_j|^2) \\ & = O(\prod_{j=0}^{i} |(\hat{t}_{i+1} - t^\star) - (\hat{t}_j - t^\star)^2|) = O(\prod_{j=0}^{i} |\hat{t}_j - t^\star|^2) \\ & = O(\prod_{j=0}^{i-1} |\hat{t}_j - t^\star|) \cdot |\hat{t}_i - t^\star|^2 = O(|\hat{t}_i - t^\star|^3), \end{aligned} \tag{21}$$

and the proof is completed. From Corollaries 1 and 2, the algorithms by computing t_i and $\hat{t}_i$

can achieve convergence order 2 and 3, respectively.

Remark 1. For a large i, the curves interpolating $i+1$ points of $\mathbf{C}(t)$ (with the order $0, 1, \cdots, i$) may cause Runge oscillation phenomenon. Thus, for $i \geq 6$, one may use three points (with the order $i-2, i-1, i$) instead for avoiding the Runge oscillation phenomenon, whose convergence orders are $p_0 \approx 1.839$ and $p_1 \approx 2.919$, which satisfy $1 + p_0 + p_0^2 - p_0^3 = 0$ and $2 + 2p_1 + 2p_1^2 - p_1^3 = 0$.

Remark 2. When a given polynomial $f(t)$ has a unique root with an interval $[a, b]$, the convergence of the corresponding progressive method by using another polynomial $a(t)$ interpolating $f(t)$ within $[a, b]$ has been discussed and proved [7, 10]. In principle, the progressive method in this paper uses a rational interpolation polynomial $G_i(s)$ to approximate $f(\phi_i(s))$ with $s \in [a, b]$, whose convergence is equivalent to that by using a rational interpolating polynomial $\bar{G}(t)$ to approximate $f(t)$, and it has similar convergence within $[a, b]$ containing a unique root.

2.5 Illustration of the RBM method for a univariate case

Table 1. Approximation errors e_k and $\hat{e}_k$ of Example 1 for different k.

k	2	3	4	5	6	7	8
e_k	1.0e-1	1.7e-3	2.3e-6	6.3e-12	4.2e-23	2.1e-45	5.3e-90
$\hat{e}_k$	1.0e-1	7.6e-5	6.7e-13	4.9e-37	1.9e-109	1.2e-326	3.1e-978

Example 1. Let $f_1(t) = (5t-2)(4-t)(t+10)^2/40, t \in [0, 1]$, which has a simple root $t^\star = 0.4 \in [0, 1]$. At the beginning, we obtain the linear function $L(s) = -10 + 23.6125s$, $t_0 = s_0 = 0$, $t_1 = s_1 = 1$ and $s^\star = 800/1889 \approx 0.4235$. And then, let $t_2 = 0.5$ and $i = 2$, we obtain $s_2 \approx 0.532224$ by using Eq. (2), and obtain $t_3 \approx 0.398328$ (or $\hat{t}_3 \approx 0.399923$) from Eq.(4) (or Eq. (10)). Later, one can obtain the values of t_i and $\hat{t}_i$ in a similar way, $i = 4, 5, \cdots$. More details of the errors $e_k = |t_k - t^\star|$ and $\hat{e}_k = |\hat{t}_k - t^\star|$ are shown in Table 1. It shows that the convergence orders for e_k and $\hat{e}_k$ are 2 and 3, respectively.

§3 Qualitative comparisons among different methods

Table 2. The symbols of different methods.

Newton like			Clipping			Progressive		
N_1	N_2[16]	N_3[23]	C_1[17]	C_2[4]	C_3[5]	M_1[RBM]	M_2[6]	M_3[7]

As shown in Table 2, the symbols of different methods of three classes are listed. In this section, the qualitative comparisons are done. All of the examples have been tested by using the Maple software on a PC with Intel i5 CPU 2*2.3G and memory 8G. The unit of the computation time is second (s).

Table 3. Comparisons with other methods using an efficiency index.

Method	N_1	N_2[16]	N_3[23]	C_1[17]	C_2[4]	C_3[5]	M_1	M_2[6]	M_3[7]
n_{fe}	2	6	4	4	7	5	1	1	1
CR	2	16	8	4	7	5	2	2	2
AEI	1.414	1.587	1.682	1.414	1.38	1.32	2	2	2

Firstly, we do the comparisons among different methods on asymptotic efficiency index (AEI), which is frequently used for comparing the computational efficiencies of different algorithms [22, 23]. The results are list in Table 3, where n_{fe}, CR and AEI denote the number n of FEs, the convergence rate p and the asymptotic efficiency index (AEI), respectively, with AEI being defined by $p^{1/n}$. In Table 3, from the second or third step, the progressive methods M_1, M_2 and M_3 achieve convergence rate 2 by adding one more FE, while both the numbers of FEs and the convergence rates of other methods are also listed. It shows that the progressive methods achieve better AEI than those of Newton-like methods and clipping methods.

Secondly, more qualitative comparisons among three classes of the methods. In principle, the progressive methods can much easier to improve the convergence order by using more derivatives than those of both Newton-like methods and clipping methods, and the computational efficiency of the progressive methods is the highest among the three classes of the methods. On the other hand, the clipping methods can be used to separate two or more roots, and it achieves the highest computational stability, and the computational stability of the progressive methods is higher than that of the Newton-like methods.

Thirdly, the comparisons among different progressive methods M_i, $i = 1, 2, 3$ are shown in Table 4. Note that their asymptotical efficiency indexes are all 2. The computational complexities of one iterative step, e.g., the n-th step, are discussed, and the corresponding results are listed in Table 4, where n_{fe}, n_a and n_m denote the number of FEs, the number of addition (minus) operations and the number of multiplication (division) operations, respectively. It shows that M_1 has better computational efficiency than those of M_2 and M_3, especially in case of a large n.

Table 4. Comparisons on computational costs for the n-th step.

Method	M_1	M_2 [6]	M_3 [7]
n_{fe}	1	1	1
n_a	$O(n)$	$O(n^2)$	$O(n^2)$
n_m	$O(n)$	$O(n^2)$	$O(n^2)$

§4 Numerical examples and further discussions

This section shows more examples for comparisons among different methods.

4.1 Comparisons between M_1 and clipping methods

This section compares M_1 with clipping methods C_2 [4] and C_3 [5]. In principle, C_2 and C_3 can compute all of the real roots of a polynomial within a given interval. For computing a unique simple root of a polynomial within an interval, M_1 can achieve a much better performance, which can be taken as a complementary one of the clipping methods. Note that C_2 [4] and C_3 [5] need 7 and 5 FEs, respectively, in each clipping step, and the length of the subinterval of the $i-$th clipping step is e_i, $i = 1, 2, \cdots$. In Table 5, the progressive method M_1 costs 5 FEs per step, i.e., the $i-$th error e_i of M_1 is mapping to $|t_{5i-1} - t^\star|$, such that there are $5i$ FEs in M_1. The average computational time is relative to the effective number of digits after point. In this paper, without special claim, the number of digits is set as 20 for testing the computational time, i.e., the tolerance is set as 10^{-20} which is enough for most of practical uses. However, the numerical convergence rate tends to be more accurate for a large number of digits, by using the Maple software, the maximum number of digits can be set up to 5000 for testing high precision of the error.

Table 5. Comparisons on errors and time of Example 2 ($|t_{5i-1} - t^\star|$ for M_1).

e_i		e_1	e_2	e_3	CR	Time(s)
$f_2(t)$	M_1	2.8e-3	9.2e-69	9.7e-2166	32	0.188
	C_2	3.0e-8	5.6e-61	1.2e-431	7	1.656
	C_3	5.1e-5	8.6e-25	1.2e-123	5	1.156
$f_3(t)$	M_1	1.7e-4	1.5e-113	3.6e-3608	32	0.188
	C_2	5.0e-7	4.4e-51	1.8e-359	7	1.672
	C_3	2.2e-4	1.0e-22	2.3e-114	5	1.282
$f_4(t)$	M_1	2.3e-6	3.4e-179	/	32	0.204
	C_2	5.2e-11	7.8e-83	1.3e-585	7	1.781
	C_3	3.5e-7	1.3e-39	1.2e-201	5	1.187
$f_5(t)$	M_1	2.1e-4	4.4e-106	1.4e-3358	32	0.203
	C_2	3.2e-7	3.4e-52	5.2e-367	7	1.703
	C_3	1.4e-4	4.6e-25	2.0e-127	5	1.219

Example 2. We have tested the following four polynomial functions $f_2(t) = (t - 1/4)(t + 3)(t + 4)^3$, $f_3(t) = (t - 1/5)(t - 3)^2(t + 5)^4$, $f_4(t) = 1/16(t - 2/5)(4 - t)(t - 10)^2$ and $f_5(t) = (t-3/5)(t+2)^2(t-4)^4$ within $[0, 1]$, which have simple roots $1/4$, $1/5$, $2/5$ and $3/5$, respectively. Table 5 shows the comparison results on both approximation error and computation time. In Table 5, Time and CR denote average computation time of one step (in second under 100 valid digits) and convergence rate, and the number 5.2e-3 denotes $5.2 \cdot 10^{-3}$. The CRs of M_1, C_2 and C_3 are 32, 7 and 5, respectively. The computational efficiency of each step of M_1 is $6 \sim 10$ times of those of C_2 and C_3. Note that e_3 of M_1 can reach 5.1e-2011 for $f_2(t)$, the bounds of the number of valid digits is set within $[16, 4000]$. It shows that M_1 achieves the best performance compared with that of C_2 and C_3.

We have also tested Example 2 under error tolerance 10^{-16} where the precision is also set as 10^{-16}. The comparison results are shown in Table 6. In Table 6, n_c denotes the number of (clipping) steps which are needed, and T_c denotes the total computation time in second. It

shows that M_1, C_2 and C_3 can satisfy the given tolerance 10^{-16} within 2 (clipping) steps, and M_1 can be done with the highest computational efficiency, which is about $8 \sim 15$ times of those of C_2 and C_3. Since M_1 is a progressive method, it can end with an earlier t_k, $k \leq 9$, which can further improve the computational efficiency.

Table 6. The number of clipping steps and computation time (s) within 10^{-16}.

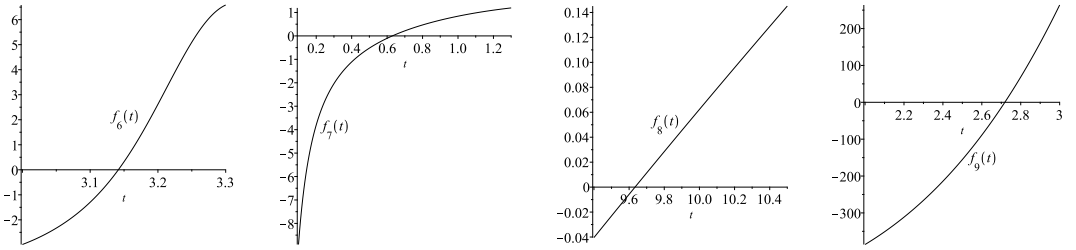
Method	$f_2(t)$		$f_3(t)$		$f_4(t)$		$f_5(t)$	
	n_c	T_c	n_c	T_c	n_c	T_c	n_c	T_c
M_1	2	0.266	2	0.272	2	0.265	2	0.281
C_2[4]	2	3.297	2	3.266	2	3.391	2	3.344
C_3[5]	2	2.516	2	2.563	2	2.485	2	2.578

4.2 Comparisons between progressive methods and Newton-like methods

This section compares progressive methods M_1, M_2 [6] and M_3 [7] with Newton-like methods N_2 [16] and N_3 [23]. Note that each iterative step of N_2 and N_3 costs 6 and 4 FEs, respectively, and each step of progressive methods is considered to cost 6 FEs. We use a total of 12 FEs for comparisons, which is composed of 2 steps of M_1, M_2, M_3 and N_2, and 3 steps of N_3.

Table 7. Comparison results on approximation errors in Example 3.

		M_1	M_2 [6]	M_3[7]	N_2 [16]	N_3[23]
$f_6(t)$	$n=4$	/	/	/	/	3.3e-2
	$n=6$	2.5e-14	1.8e-18	3.9e-13	6.6e-5	/
	$n=8$	/	/	/	/	1.2e-6
	$n=12$	5.8e-825	1.6e-1076	3.7e-748	2.0e-56	3.5e-42
	CR	59	60	58	10	19
$f_7(t)$	$n=4$	/	/	/	/	5.8e-2(tb)
	$n=6$	1.8e-17	1.2e-17	7.1e-19	6.7(ta)	/
	$n=8$	/	/	/	/	8.1e-11(tb)
	$n=12$	4.2e-1026	3.1e-1109	1.7e-1202	5.2e-1(ta)	3.0e-81(tb)
	CR	60	65	63	/(diverge)	22
$f_8(t)$	$n=4$	/	/	/	/	3e-16
	$n=6$	4.1e-52	6.5e-44	2.7e-56	7e-32	/
	$n=8$	/	/	/	/	4e-136
	$n=12$	1.4e-3382	4.3e-2844	2.0e-3623	4e-520	2e-1094
	CR	65	64	64	16	22.9
$f_9(t)$	$n=4$	/	/	/	/	1.9e-2
	$n=6$	-2.4e-9	6.8e-12	4.9e-14	2.7e-5	/
	$n=8$	/	/	/	/	1.1e-16
	$n=12$	-1.4e-495	5.84e-732	3.4e-832	-2.2e-79	1.3e-130
	CR	55	61	59	16	8



(a) (b) (c) (d)

Figure 2. Plots of $f_i(t)$ in Example 3, $i = 6, 7, 8, 9$.

Table 8. Comparisons on computation time at the end of 12 FEs in Example 3.

	Digits=100				Digits=1000			
	$f_6(t)$	$f_7(t)$	$f_8(t)$	$f_9(t)$	$f_6(t)$	$f_7(t)$	$f_8(t)$	$f_9(t)$
M_1	0.453	0.391	0.437	0.484	0.797	0.703	0.750	0.758
M_2[6]	0.641	0.672	0.703	0.721	1.203	1.313	1.375	1.346
M_3[7]	0.578	0.547	0.562	0.594	0.906	0.922	0.937	0.953
N_2	0.373	0.360	0.375	0.328	0.507	0.513	0.506	0.375
N_3	0.391	0.375	0.390	0.391	0.532	0.542	0.537	0.422

Example 3. We have compared progressive methods M_1, M_2 and M_3 with Newton-like methods N_2 and N_3, by testing three non-polynomial functions (see also Fig. 2),

$$\begin{aligned}
&f_6(t) = (t-0.5)(e^{\sin(10(t-\pi))} + 4(t-\pi) - 1), && t \in [3, 3.3], && t_2^\star = \pi,\\
&f_7(t) = -1/t + \sin(t) + 1, && t \in [0.01, 1.3], && t_3^\star \approx 0.6294465,\\
&f_8(t) = \sqrt{t} - 1/t - 3, && t \in [9.4, 10.5], && t_4^\star \approx 9.6318875,\\
&f_9(t) = 10t^4 - \ln(t) - 10e^4 + 1, && t \in [2, 3], && t_5^\star = e,
\end{aligned}$$

which have simple roots $t_i^\star$, $i = 2, 3, 4, 5$, respectively. With an initial value, by using Theorem 3, one can obtain an interval containing the root $t_i^\star$. Tables 7 and 8 show the comparison results on approximation error and average computation time of each step. N_2 and N_3 converge to the correct result for cases $f_6(t)$ and $f_7(t)$; while for the $f_8(t)$ case, N_2 diverges around $t_a = 5.3347$, and N_3 converges to a wrong one $t_b = 16.933$. The progressive methods converge to the correct results for all three cases. Table 7 shows the comparison on the approximation errors of the progressive methods and the Newton-like method. As shown in Table 7, the progressive methods M_1, M_2 and M_3 have comparable convergence rates, and each additional FE can achieve twice the convergence order, which are much better than those of Newton-like methods N_2 and N_3.

Table 8 shows the total computation time at the end of 12 FEs under different digits as well, which shows that M_1, N_2 and N_3 achieve comparable computational efficiency, which are better than those of M_2 and M_3. M_1 achieves the best approximation error in the same time, and achieves the best computational efficiency for the same approximation tolerance. Considering all the above comparisons, M_1 achieves the best performance among the five methods M_1, M_2, M_3, N_2 and N_3.

Example 4. We also have tested the following polynomial function $f_{10}(t) = 10t^4 + \log(2 -$

$t), t \in [2-10^{-69}, 2-10^{-70}]$. Table 9 shows that M_1 works well, while Newton method reaches $tc = 2+10^{-70}$ which is outside of the valid domain of definition in the first iteration. In this case, M_1 achieves a better robustness.

Table 9. Comparison results on approximation errors in Example 4 ($e_k = |t_k - t^\star|$).

e_k	e_2	e_4	e_6	e_8	e_{10}	e_{12}
M_1	2.3e-70	4.5e-71	1.5e-75	5.4e-94	4.3e-168	2.1e-301
Newton	tc	/	/	/	/	/

4.3 Further discussions with multiple roots for a univariate case

Firstly, we show how to obtain an interval containing one root. Suppose that x_1 is obtained by using some iterative methods including the Newton's method under the initial value x_0. If $2|x_1 - t^\star| < |x_0 - t^\star|$, where $t^\star$ is the root of $f(t)$, from Lemma 2, $t^\star$ is bounded by x_0 and $2x_1 - x_0$. In this way, one can find an interval containing a root by using a suitable initial value x_0.

Secondly, we consider the case which contains a multiple root $t^\star$ within an interval, and $k \geq 2$ is the corresponding multiplicity. If k is even, we utilize $\mathbf{A}_i(s) = (s/g_i(s), L(s)/g_i(s))$ to approximate $\mathbf{C}_1(t) = (f(t), f'(t))$. Otherwise, if k is odd, we utilize $\mathbf{A}_i(s)$ to approximate $\mathbf{C}_2(t) = (f'(t), f''(t))$ instead. In principle, $(f^{(k-2)}(t), f^{(k-1)}(t))$ is the best one to be approximated by using $\mathbf{A}_i(s)$, which can lead to a better convergence rate.

Thirdly, the case which may contain two or more roots is considered. If the given function is a polynomial, one possible way is to turn it into Bernstein-Bézier form and split the given interval into several subintervals by using the zeros of its control polygon. For each subinterval, one may isolate the roots by using the method of the first case. For a non-polynomial case, one may sample the interval into several subintervals of smaller length, and use the method of the first case to iteratively split it into several subintervals which contain one root.

Example 5. By using the methods for the above three cases, we have tested the RBM method M_1 to compute the roots of the Wilkinson polynomial (see also Fig. 3),

$$W(t) = \prod_{i=0}^{20}(t-i), \tag{22}$$

within $[0, 25]$, which has twenty zeros $i, i = 1, 2, \cdots 20$, see also Fig. 1 and Example 7 in [4]. At the beginning, we compute the zeros of the corresponding control polygon, i.e., $\{0.27, 1.55, 2.83, 4.11, 5.38, 6.65, 7.92, 9.19, 10.46, 11.73, 13.007, 14.27, 15.54, 16.81, 18.07, 19.34, 20.61, 21.87, 23.14, 24.40\}$. Thus, the given interval [0, 25] is divided into twenty-two sub-intervals by using the above twenty-one zeros. There are sixteen sub-intervals containing one or two roots of $W(t)$. We select three of them $\bigwedge_1 = [0.27, 1.55]$, $\bigwedge_2 = [2.83, 4.11]$ and $\bigwedge_3 = [16.81, 18.07]$ to illustrate with more details, which contain one, two and two roots of $W(t)$, respectively. The RBM can be directly applied to $\bigwedge_1$ where $W(t)$ has different signs at its two end points. For $\bigwedge_2$ and $\bigwedge_3$, one can select their mid points to split each of them into two subintervals and each subinterval con-

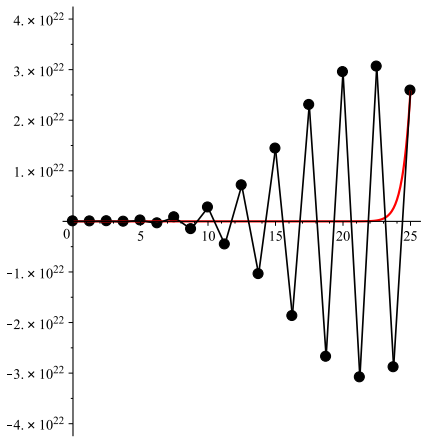


Figure 3. Plot of the Wilkinson polynomial [4].

tains a root. The RBM can then be applied for the subintervals. By applying the optimization method of the given interval (see also Remark 1) for adding one or two points, one may obtain optimized intervals $\bar{\Lambda}_1 = [0.99, 1.0034]$, $\bar{\Lambda}_2 = [2.83, 3.004]$ and $\bar{\Lambda}_3 = [17.988, 18.07]$ which are corresponding to $[0.27, 1.55]$, $[2.83, 3.47]$ and $[17.44, 18.07]$. As shown in Table 10, the RBM works well in these cases, where the error e_i is mapping, respectively to the reparameterization function $\phi_i(t)$.

Table 10. The approximation errors of M_1 in Example 5.

Error	e_1	e_2	e_3	e_4	e_5	e_6	e_7
$\bar{\Lambda}_1$	2.9e-5	7.7e-9	3.7e-15	2.5e-28	2.6e-54	1.7e-106	1.0e-210
$\bar{\Lambda}_2$	4.8e-4	1.2e-6	1.2e-11	1.2e-21	1.2e-41	1.2e-81	1.3e-161
$\bar{\Lambda}_3$	2.8e-3	5.6e-5	1.4e-7	3.2e-14	2.9e-25	5.4e-49	2.0e-95

4.4 Example of the RBM by using derivatives

Table 11. Approximation errors e_i of Example 6 for different i.

e_i	3	4	5	6	7	CR
$f_{11}(t)$	2.6e-8	4.7e-23	2.0e-67	2.5e-202	7.2e-603	3
$f_{12}(t)$	1.3e-3	2.6e-5	2.5e-10	2.1e-25	1.4e-70	3
$f_{13}(t)$	3.5e-6	1.2e-17	9.3e-50	5.7e-147	6.9e-439	3
$f_{14}(t)$	1.8e-5	8.0e-15	2.4e-43	1.0e-128	9.3e-385	3
$f_{15}(t)$	2.4e-9	1.1e-28	1.1e-86	1.2e-260	1.5e-782	3
$f_{16}(t)$	2.0e-6	2.0e-18	1.4e-53	4.9e-160	4.6e-478	3

Example 6. We have tested the RBM with derivatives by using the following eight more

examples,

$$\begin{array}{llll} f_{11}(t)=(t-1)^3-1, & t^\star=2, & t_0=1.9, & t_1=2.011,\\ f_{12}(t)=10^{150-5t^2}-1, & t^\star\approx 5.4772, & t_0=5.490, & t_1=5.458,\\ f_{13}(t)=e^{-t^2+t+3}-t+2, & t^\star\approx 2.490, & t_0=2.3, & t_1=2.6,\\ f_{14}(t)=t^3-10, & t^\star=10^{1/3}, & t_0=2, & t_1=2.850,\\ f_{15}(t)=\sqrt{t}-1/t-3, & t^\star\approx 9.6335, & t_0=9.4, & t_1=10.5,\\ f_{16}(t)=e^t+t-20, & t^\star\approx 2.8424, & t_0=2.5, & t_1=3. \end{array}$$

The results are shown in Table 11. It shows that RBM by using derivatives works well and achieves the convergence rate 3, and it means that RBM by using derivatives can achieve much better convergence order for each additional FE.

§5 Extending the RBM method to bivariate cases

5.1 The algorithm of the RBM method

Given an equation system with $x\in[a,b], y\in[c,d]$ as follows

$$\begin{cases} f_1(x,y)=0,\\ f_2(x,y)=0, \end{cases} \tag{23}$$

which has a solution $(x^\star,y^\star)\in[a,b]\times[c,d]$, and the initial value is (x_0,y_0).

Let $L_i(x,y)=f_i(x_0,y_0)+f_{ix}(x_0,y_0)(x-x_0)+f_{iy}(x_0,y_0)(y-y_0)\triangleq\alpha_i x+\beta_i y+\mu_i$, where f_{ix} and f_{iy} denote the partial derivatives of $f_i(x,y)$ in x and y, respectively, and $s^\star\in[a,b], t^\star\in[c,d]$ satisfy $L_i(s^\star,t^\star)=0$, $i=1,2$. In this case, we find curves $\frac{(s,L_1(s,t))}{u_i(s,t)}$ and $\frac{(t,L_2(s,t))}{v_i(s,t)}$ to interpolate $(x,f_1(x,y))$ and $(y,f_2(x,y))$ at three points (x_j,y_j), $j=i-2,i-1,i$, where $i\geq 2$,

$$u_i(s,t)=a_{i,0}+a_{i,1}s+a_{i,2}t \quad\text{and}\quad v_i(s,t)=b_{i,0}+b_{i,1}s+b_{i,2}t \tag{24}$$

are linear functions of both s and t.

Given three points (x_j,y_j), $j=i-2,i-1,i$, the next point can be similarly computed in the following way. Firstly, for each j, the values of s_j and t_j are computed by solving the system consisting of the following two linear equations

$$\begin{cases} (s_j,L_1(s_j,t_j))\times(x_j,f_1(x_j,y_j))=f_1(x_j,y_j)s_j-L_1(s_j,t_j)x_j=0,\\ (t_j,L_2(s_j,t_j))\times(y_j,f_2(x_j,y_j))=f_2(x_j,y_j)t_j-L_2(s_j,t_j)y_j=0. \end{cases} \tag{25}$$

Secondly, the values of $a_{i,j}$ and $b_{i,j}$, $j=0,1,2$, are computed by solving

$$\begin{cases} u_i(s_j,t_j)=a_{i,0}+a_{i,1}s_j+a_{i,2}t_j=\frac{s_j}{x_j},\\ v_i(s_j,t_j)=b_{i,0}+b_{i,1}s_j+b_{i,2}t_j=\frac{t_j}{y_j}, \end{cases} \quad j=i-2,i-1,i. \tag{26}$$

Thirdly, the values of x_{i+1} and y_{i+1} are computed by using

$$x_{i+1}=\frac{s^\star}{u_i(s^\star,t^\star)} \quad\text{and}\quad y_{i+1}=\frac{t^\star}{v_i(s^\star,t^\star)}. \tag{27}$$

The outline of the algorithm for a bivariate case is described as follows.

Algorithm 2. The RBM method for solving the solution $(x^\star,y^\star)$ of Eq. (23).

Input: The given smooth functions $f_i(x,y)$, the intervals $[a,b]$ and $[c,d]$, the initial value (x_0,y_0) and the tolerance ε.

Output: The approximation of $(x^\star,y^\star)$.

(1) **Begin:** At the beginning, given (x_0, y_0), the points (x_1, y_1) and (x_2, y_2) can be sampled from the circle with the center (x_0, y_0) and its radius 10ε;
Compute $(s^\star, t^\star)$ by solving $L_1(s,t) = L_2(s,t) = 0$;
Compute (s_0, t_0), (s_1, t_1) by solving Eq. (25);
Let $i = 2$.

(2) If $\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} < \varepsilon$, go to Step ((7)); otherwise, go to Step ((3)).

(3) Compute (s_i, t_i) by solving Eq. (25);

(4) By solving Eq. (26), one obtains $u_i(s,t)$ and $v_i(s,t)$.

(5) By using Eq. (27), one obtains the values of x_{i+1} and y_{i+1}.

(6) By setting $i = i + 1$, go to Step (2).

(7) **End:** Output (x_i, y_i) as the approximation of $(x^\star, y^\star)$.

5.2 Numerical examples

Example 7. We have tested the RBM method by using the bivarite cases in Table 12, which are shown in Fig. 4.

Table 12. The bivarite cases for testing the RBM method

Example	formula	initial value	root
EX_1	$\begin{array}{l}(y-1)^3(x+1)+yx^3-4=0\\ (x+1)^2-2y^2+4x=0\end{array}$	$x_0=0.7; y_0=1.6$	$x^\star=1; y^\star=2$
EX_2	$\begin{array}{l}ln(2e^{x^2-y}-xy)+e^x-e^y=0\\ (x-2)^4(y+1)^{xy}+y^{\ln(x)}-3=0\end{array}$	$x_0=0.8; y_0=0.8$	$x^\star=1; y^\star=1$
EX_3	$\begin{array}{l}(x^4-1)^2(y^2-2)-x^2y^2-3=0\\ e^{y^2-x^2-1}-(x^2-y^2)^2=0\end{array}$	$x_0=1.35; y_0=1.83$	$x^\star=\sqrt{2}; y^\star=\sqrt{3}$
EX_4	$\begin{array}{l}3x^3-4x^2+y^4-3=0\\ \ln(x)-y^2+2=0\end{array}$	$x_0=0.85; y_0=1.65$	$x^\star=1; y^\star=\sqrt{3}$
EX_5	$\begin{array}{l}y\ln(x)+y^{\frac{1}{2}(x-3)^2}-2=0\\ \ln(x^y y^x)-y^{\ln(x)}=0\end{array}$	$x_0=2.5; y_0=1.2$	$x^\star=e; y^\star=1$
EX_6	$\begin{array}{l}x^3-x-0.1+y=0\\ \cos(\frac{1}{y})+e^y-\cos(10)-e^{0.1}+1-x=0\end{array}$	$x_0=0.9; y_0=0.08$	$x^\star=1; y^\star=0.1$

By using Algorithm 2, we obtain the results shown in Table 13. Note that RBM costs one functional evaluation (i.e., $f(x,y)$) for each step while the Newton's method costs three functional evaluations (i.e., $f(x,y)$, $f_x(x,y)$ and $f_y(x,y)$) for each step. In Table 12, both the approximation error and the computational time are of three steps for RBM, while they are of one step for the Newton's method. It shows that the convergence rate of RBM is about 3 for three functional evaluations (or three steps) or $3^{1/3} \approx 1.44$ for one functional evaluation

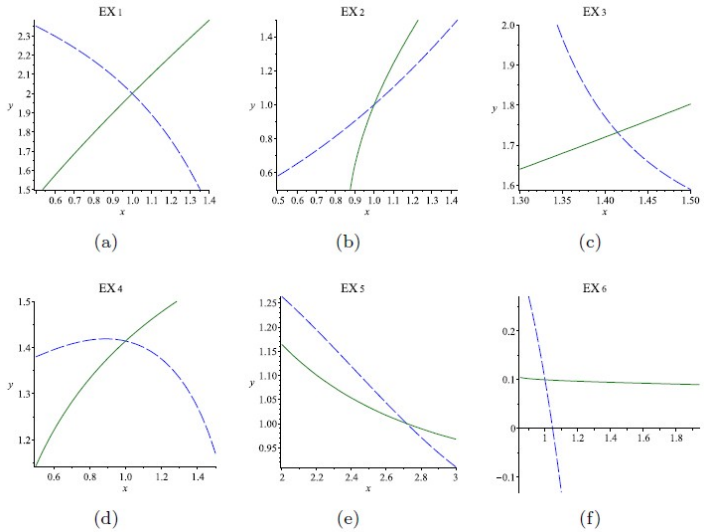


Figure 4. Example 7. Plots of curves in cases EX_i, $i = 1, 2, \cdots, 6$.

(or one step), while that of the Newton's method is of convergence rate 2 for three functional evaluations or $2^{1/3} \approx 1.25$ for one functional evaluation. The computational time of RBM for one step is very close to one-third of that of the Newton's method, whose unit is second. Both of RBM and the Newton's method work well for the first five examples; for the sixth example, RBM works well while the Newton's method diverges. It shows that the performance of RBM is much better than that of the Newton's method.

Table 13. Comparisons on errors and computational time of Example 7.

Example	Method	1	2	3	4	Time(s)	CR
EX_1	RBM	7.6e-3	5.5e-9	4.4e-28	4.7e-89	0.094	3.18
	Newton	1.1e-2	1.1e-4	1.1e-8	1.2e-16	0.094	2
EX_2	RBM	1.2e-2	1.4e-8	2.8e-25	5.1e-80	0.094	3.20
	Newton	7.4e-6	4.8e-11	2.1e-21	3.8e-42	0.095	2
EX_3	RBM	2.2e-3	8.9e-10	1.5e-29	5.5e-92	0.095	3.17
	Newton	5.8e-7	5.0e-13	3.9e-25	2.2e-49	0.094	2
EX_4	RBM	4.0e-3	5.4e-11	1.3e-35	3.0e-112	0.094	3.20
	Newton	9.7e-5	9.4e-9	8.9e-17	8.0e-33	0.096	2
EX_5	RBM	3.6e-3	3.8e-9	5.8e-28	1.3e-89	0.094	3.18
	Newton	1.7e-3	2.8e-6	7.2e-12	4.8e-23	0.096	2
EX_6	RBM	1.6e-3	3.5e-9	7.0e-26	2.2e-78	0.096	3
	Newton	1.49	1.61	1.86	1.86	0.094	/(diverge)

§6 Discussions and conclusions

Numerical examples show that the RBM method can achieve convergence rate 2 or 3 by using no derivatives or with derivatives, the corresponding asymptotical efficiency indexes are 2 or $\sqrt{3}$, which are much better than that $\sqrt{2}$ of the Newton's method. Furthermore, note that the interpolation result doesn't matter with the order of the interpolation points, while the Newton's method does matter with the order of the point sequence, and the RBM method can work more robust than the Newton's method.

Comparing with the progressive methods M_2 [6] and M_3 [7], it achieves much higher computational efficiency, while the approximation error is equivalent to those of M_2 and M_3. Moreover, M_2 and M_3 cannot deal with the bivariate cases, while M_1 is extended to bivariate cases.

In our future work, it is challenging to extend the RBM method to solve double or more roots with the same number of initial values at the same time. Next, it is necessary to further improve the corresponding robustness. Finally, it is meaningful to self-adaptively find suitable initial values for RBM, and to extend RBM to solve equation systems of three or more variables.

Acknowledgement

The authors would like to thank anonymous reviewers for their helpful comments and suggestions.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

[1] M Aizenshtein, M Bartoň, G Elber. *Global solutions of well-constrained transcendental systems using expression trees and a single solution test*, Computer Aided Geometric Design, 2012, 29(5): 265-279.

[2] M Bartoň, B Jüttler. *Computing roots of polynomials by quadratic clipping*, Computer Aided Geometric Design, 2007, 24(3): 125-141.

[3] M Bartoň, G Elber, I Hanniel. *Topologically guaranteed univariate solutions of underconstrained polynomial systems via no-loop and single-component tests*, Computer-Aided Design, 2011, 43(8): 1035-1044.

[4] X Chen, W Ma, Y Ye. *A rational cubic clipping method for computing real roots of a polynomial*, Computer Aided Geometric Design, 2015, 38: 40-50.

[5] X Chen,W Ma. *Rational cubic clipping with linear complexity for computing roots of polynomials*, Applied Mathematics and Computation, 2016, 273: 1051-1058.

[6] X Chen, Y Zhang, J Shi, et al. *An efficient method based on progressive interpolation for solving non-linear equations*, Applied Mathematics Letters, 2016, 61 (11): 67-72.

[7] X Chen, J Shi, W Ma. *A fast and robust method for computing real roots of nonlinear equations*, Applied Mathematics Letters, 2017, 68: 27-32.

[8] Y Choi, W Wang, Y Liu, et al. *Continuous collision detection for two moving elliptic disks*, IEEE Transactions on Robotics, 2006, 22(2): 213-224.

[9] A Cordero, J Torregrosa, M Vassileva. *A family of modified Ostrowski's methods with optimal eighth order of convergence*, Applied Mathematics Letters, 2011, 24(12): 2082-2086.

[10] P Davis. *Interpolation and Approximation*, New York: Dover Publications, 1975.

[11] J Džunić, I Damnyanović. *General approach to constructing optimal multipoint families of iterative methods using Hermite's rational interpolation*, Journal of Computational and Applied Mathematics, 2017, 321: 261-269.

[12] G Elber, M S Kim. *Geometric constraint solver using multivariate rational spline functions*, In: Proceedings of the sixth ACM Symposium on Solid Modeling and Applications, Michigan, Ann Arbor, USA, 2001, 1-10.

[13] R Farouki, T Goodman. *On the optimal stability of the Bernstein basis*, Mathematics of Computation, 1996, 65(216): 1553-1566.

[14] B Jüttler. *The dual basis functions of the Bernstein polynomials*, Advances in Computational Mathematics, 1998, 8: 345-352.

[15] H Kung, J Traub. *Optimal order of one-point and multi-point iteration*, Applied Mathematics and Computation, 1974, 21(4): 643-651.

[16] X Li, C Mu, J Ma, et al. *Sixteenth-order method for nonlinear equations*, Applied Mathematics and Computation, 2010, 215(10): 3754-3758.

[17] L Liu, L Zhang, B Lin, et al. *Fast approach for computing roots of polynomials using cubic clipping*, Computer Aided Geometric Design,2009, 26(5): 547-559.

[18] K Morken, M Reimers. *An unconditionally convergent method for computing zeros of splines and polynomials*, Mathematics of Computation, 2007, 76(258): 845-865.

[19] R E Moore. *Methods and applications of interval analysis*, Philadelphia: Society for Industrial and Applied Mathematics, 1979.

[20] R Moore, R Kearfott, M Cloud. *Introduction to interval analysis*, Philadelphia: Society for Industrial and Applied Mathematics, 2009.

[21] T Sederberg, T Nishita. *Curve intersection using Bézier clipping*, Computer-Aided Design, 1990, 22(9): 538-549.

[22] J R Sharma, H Arora. *An efficient family of weighted-Newton methods with optimal eighth order convergence*, Applied Mathematics Letters, 2014, 29(1): 1-6.

[23] J R Sharma, H Arora. *A new family of optimal eighth order methods with dynamics for nonlinear equations*, Applied Mathematics and Computation, 2016, 273(1): 924-933.

[24] I Petkovic, D Herceg. *Computers in mathematical research: The study of three-point root-finding methods*, Numerical Algorithms, 2020, 84(3): 1179-1198.

[25] R Behl, S Amat, A Magrenan, et al. *An efficient optimal family of sixteenth order methods for nonlinear models*, Journal of Computational and Applied Mathematics, 2019, 354: 271-285.

[26] I Pavaloiu. *On an Aitken-Steffensen-Newton type method*, Carpathian Journal of Mathematics, 2018, 34(1): 85-92.

[27] R Behl, A Cordero, S Motsa, et al. *An eighth-order family of optimal multiple root finders and its dynamics*, Numerical Algorithms, 2018, 77(4): 1249-1272.

[28] R Behl, D. Gonzalez, P Maroju, et al. *An optimal and efficient general eighth-order derivative free scheme for simple roots*, Journal of Computational and Applied Mathematics, 2018, 330: 666-675.

[29] S Li, G Chen, Y Wang. *An improved rational cubic clipping method for computing real roots of a polynomial*, Applied Mathematics and Computation, 2019, 349: 207-213.

[30] C Chun, B Neta. *Comparative study of methods of various orders for finding simple roots of nonlinear equations*, Journal of Applied Analysis and Computation, 2019, 9(2): 400-427.

[31] O Solaiman, S Karim, I Hashim. *Optimal fourth- and eighth-order of convergence derivative-free modifications of King's method*, Journal of King Saud University - Science, 2019, 31(4): 1499-1504.

[32] D Herceg, D Herceg. *Eighth order family of iterative methods for nonlinear equations and their basins of attraction*, Journal of Applied Mathematics and Computing, 2018, 343: 458-480.

[33] S Sariman, I Hashim. *New optimal Newton-Householder methods for solving nonlinear equations and their dynamics*, Computers, Materials and Continua, 2020, 65(1): 69-85.

Zhejiang Key Laboratory of New Industrial Internet Control Technology, Hangzhou Dianzi University, Hangzhou 310018, China.

Emails: 277179337@qq.com, 690106853@qq.com, xiaodiao@hdu.edu.cn