AIMS Mathematics, 9(10): 28623–28642.
DOI: 10.3934/math.20241389
Received: 01 August 2024
Revised: 21 September 2024
Accepted: 26 September 2024
Published: 09 October 2024

https://www.aimspress.com/journal/Math

Research article

A multi-step Ulm-Chebyshev-like method for solving nonlinear operator equations

Wei Ma[1,*], **Ming Zhao**[2] **and Jiaxin Li**[1]

[1] School of Mathematics and Statistics, Nanyang Normal University, Nanyang 473061, China

[2] Public Basic Teaching Department, Henan Police College, Zhengzhou 450000, China

* **Correspondence:** Email: 20131105@nynu.edu.cn; Tel: +8615036280979.

Abstract: In this paper, based on the Ulm-Chebyshev iterative procedure, we present a multi-step Ulm-Chebyshev-like method to solve systems of nonlinear equations $F(x) = 0$,

$$\begin{cases} \mathbf{y}_n = \mathbf{x}_n - B_n F(\mathbf{x}_n), \\ \mathbf{z}_n = \mathbf{y}_n - B_n F(\mathbf{y}_n), \\ \mathbf{x}_{n+1} = \mathbf{z}_n - B_n F(\mathbf{z}_n), \\ \bar{B}_n = 2B_n - B_n A_{n+1} B_n, \\ B_{n+1} = \bar{B}_n + \bar{B}_n(2I - A_{n+1}\bar{B}_n)(I - A_{n+1}\bar{B}_n), \quad n = 0, 1, 2, \ldots, \end{cases}$$

where A_{n+1} is an approximation of the derivative $F'(\mathbf{x}_{n+1})$. This method does not contain inverse operators in its expression, and does not require computing Jacobian matrices for solving Jacobian equations. We have proved that the multi-step Ulm-Chebyshev-like method converges locally to the solution with R-convergence rate 4 under appropriate conditions. Some applications are given, compared with other existing methods, where the most important features of the method are shown.

Keywords: nonlinear equation; multi-step; Ulm-Chebyshev-like method; R-convergence rate 4
Mathematics Subject Classification: 47H30, 65H10, 65J15

1. Introduction

Let X and Y be Banach spaces, $D \in X$ be an open subset, and F: $D \in X \to Y$ be a nonlinear operator with the continuous Fréchet derivative denoted by F'. We consider the problem of approximating a solution $\mathbf{x}^*$ of a nonlinear equation

$$F(\mathbf{x}) = 0, \tag{1.1}$$

which applies to the inverse eigenvalue problems [1–4], the generalized inverse eigenvalue problems [5], the inverse singular value problems [6–8], the Chandrasekhar integral equation [9], the neutral differential-algebraic equations [10, 11], and so on. Without any doubt, Newton's method is the most-used iterative process to solve this problem. It is given by the algorithm:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - F'(\mathbf{x}_n)^{-1}F(\mathbf{x}_n), \quad n \geq 0$$

for $\mathbf{x}_0$ given. This iterative process has quadratic R-order of convergence under some mild conditions [12–14]. For recent progress on Newton's method, one may refer to [15–17].

Other methods, such as higher-order methods, also include in their expression the inverse of the operator F'. To avoid this problem, Newton-type methods:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - H_n F(\mathbf{x}_n),$$

where H_n is an approximation of $F'(\mathbf{x}_n)^{-1}$ are considered. One of these methods was proposed by Moser in [18]. Given $\mathbf{x}_0 \in D$ and $B_0 \in \mathcal{L}(Y, X)$, the following sequences are defined:

$$\begin{cases} \mathbf{x}_{n+1} = \mathbf{x}_n - B_n F(\mathbf{x}_n), \\ B_{n+1} = 2B_n - B_n F'(\mathbf{x}_n) B_n, \quad n = 0, 1, 2, \ldots. \end{cases} \tag{1.2}$$

The first equation is similar to Newton's method, but replaces the operator $F'(\mathbf{x}_n)^{-1}$ with a linear operator B_n. The second equation is Newton's method applied to the equation

$$g_n = 0$$

where g_n: $\mathcal{L}(Y, X) \to \mathcal{L}(X, Y)$ is defined by

$$g_n(B) = B^{-1} - F'(\mathbf{x}_n).$$

So $\{B_n\}$ gives us an approximation of $F'(\mathbf{x}_n)^{-1}$. It can be shown that the rate of convergence for the above scheme is $(1 + \sqrt{5})/2$, provided the root of (1.1) is simple [18]. However, from a numerical perspective, this is unsatisfactory because the scheme uses the same amount of information in each step as Newton's method, but its convergence speed is not faster than the secant method. For that, in [19], Ulm proposed the following iterative method to solve nonlinear equations. Given $\mathbf{x}_0 \in D$ and $B_0 \in \mathcal{L}(Y, X)$, Ulm defines

$$\begin{cases} \mathbf{x}_{n+1} = \mathbf{x}_n - B_n F(\mathbf{x}_n), \\ B_{n+1} = 2B_n - B_n F'(\mathbf{x}_{n+1}) B_n, \quad n = 0, 1, 2, \ldots. \end{cases} \tag{1.3}$$

Notice that, here $F'(\mathbf{x}_{n+1})$ appears instead of $F'(\mathbf{x}_n)$ in (1.2). This is crucial for obtaining fast convergence. Under the classical assumption that the derivative F' is Lipschitz continuous around the solution, Ulm showed that the method generates successive approximations that converge to a solution of (1.1) asymptotically as fast as Newton's method. For recent progress on Newton-Moser type method [20, 21], one may refer to Moser's method [22], Ulm's method [23–26], and Ulm-like method [27, 28].

Considering the previous antecedent, in order to extend the above ideas to high-order convergent iterative methods (cubic convergence), in [29], Ezquerro and Hernández considered Chebyshev's method and proposed the following iterative method to solve nonlinear equations. Given $\mathbf{x}_0 \in D$ and $B_0 \in \mathcal{L}(Y, X)$, Ulm-Chebyshev defines

$$\begin{cases} \mathbf{y}_n = \mathbf{x}_n - B_n F(\mathbf{x}_n), \\ \mathbf{x}_{n+1} = \mathbf{y}_n - B_n F(\mathbf{y}_n), \\ B_{n+1} = B_n + B_n(2I - F'(\mathbf{x}_{n+1})B_n)(I - F'(\mathbf{x}_{n+1})B_n), \quad n = 0, 1, 2, \ldots, \end{cases} \tag{1.4}$$

which does not use any inverse operator in its application. Ezquerro and Hernández showed that the method generates successive approximations that converge to a solution of (1.1), and has cubical convergence. Recently, some authors have employed Ulm-Chebyshev's method to solve inverse eigenvalue problems and inverse singular value problems [1, 2, 5, 7]. There, they found that computing exactly the derivative $F'(\mathbf{x}_n)$ $(n = 0, 1, 2, \ldots)$ at each iteration is costly, especially in the case when the system is large.

In order to reduce the cost of accurately calculating the derivative $F'(\mathbf{x}_n)$ $(n = 0, 1, 2, \ldots)$ in each iteration, motivated by the Ulm and Ulm-Chebyshev methods, we propose a multi-step Ulm-Chebyshev-like method for solving the nonlinear operator equation

$$F(\mathbf{x}) = 0.$$

Given $\mathbf{x}_0 \in D$ and $B_0 \in \mathcal{L}(Y, X)$, the multi-step Ulm-Chebyshev-like method is defined by

$$\begin{cases} \mathbf{y}_n = \mathbf{x}_n - B_n F(\mathbf{x}_n), \\ \mathbf{z}_n = \mathbf{y}_n - B_n F(\mathbf{y}_n), \\ \mathbf{x}_{n+1} = \mathbf{z}_n - B_n F(\mathbf{z}_n), \\ \bar{B}_n = 2B_n - B_n A_{n+1} B_n, \\ B_{n+1} = \bar{B}_n + \bar{B}_n(2I - A_{n+1}\bar{B}_n)(I - A_{n+1}\bar{B}_n), \quad n = 0, 1, 2, \ldots, \end{cases} \tag{1.5}$$

where A_{n+1} is an approximation of the derivative $F'(\mathbf{x}_{n+1})$. This method exhibits several attractive features. First, it is inverse free: we do not need to solve a linear equation at each iteration. Second, it is derivative free: we do not need to computer the Fréchet derivative at each iteration. Third, in addition to solving the nonlinear Eq (1.1), the method produces successive approximations $\{B_n\}$ to the value of $F'(\mathbf{x}^*)^{-1}$, having $\mathbf{x}^*$ as a solution of (1.1). This property is very helpful, especially when one investigates the sensitivity of the solution to small perturbations. Fourth, the method converges to the solution with R-convergence rate 4.

Further more, in Section 2, we analyze the local convergence of the new iterative method. Under certain assumptions, the radius of the convergence ball for the multi-step Ulm-Chebyshev-like method is estimated, and the R-convergence rate 4 of the multi-step Ulm-Chebyshev-like method is proved. Section 3 is devoted to showing some of the most important features of the new iterative method by means of five examples. Compared with other existing methods, the proposed method has higher convergence order and/or requires less operations.

2. Convergence analysis

Let $\mathbf{B}(\mathbf{x}, r)$ stand for the open ball in X with center $\mathbf{x}$ and radius $r > 0$. Let $\mathbf{x}^* \in D$ be a solution of the nonlinear Eq (1.1) such that $F'(\mathbf{x}^*)$ is invertible and that F' satisfies the Lipschitz condition on $\mathbf{B}(\mathbf{x}^*, r)$ with the Lipschitz constant L:

$$\|F'(\mathbf{x}) - F'(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \quad \text{for } \mathbf{x}, \mathbf{y} \in \mathbf{B}(\mathbf{x}^*, r). \tag{2.1}$$

Let $\{\mathbf{x}_n\}$ be generated by the multi-step Ulm-Chebyshev-like method. Let A_n be an approximation of the derivative $F'(\mathbf{x}_n)$ such that

$$\|A_n - F'(\mathbf{x}_n)\| \leq \eta_n \|F(\mathbf{x}_n)\|, \quad n = 0, 1, 2, \ldots, \tag{2.2}$$

where $\{\eta_n\}$ is a nonnegative-valued sequence satisfying $\sup_{n\geq 0} \eta_n \leq \eta$ where η is a nonnegative constant. Let

$$0 < r_L < \min\Big\{1,\ r,\ \frac{1}{(\frac{L}{2}\eta + L + \eta\|F'(\mathbf{x}^*)\|)\|F'(\mathbf{x}^*)^{-1}\|}\Big\}, \tag{2.3}$$

$$\mu = \frac{\|F'(\mathbf{x}^*)^{-1}\|}{1 - (\frac{L}{2}\eta + L + \eta\|F'(\mathbf{x}^*)\|)\|F'(\mathbf{x}^*)^{-1}\|r_L},$$

$$\gamma_1 = \eta(\frac{L}{2} + \|F'(\mathbf{x}^*)\|), \quad \gamma_2 = 1 + 2\mu\gamma_1, \quad \gamma_3 = \gamma_2 + L\mu,$$

$$\gamma_4 = \gamma_2 + 2\mu L + 2\mu L\gamma_3, \quad \gamma_5 = \gamma_4 + 2\mu L + 2\mu L\gamma_3, \quad \gamma_6 = 1 + 4\mu\gamma_1 + 4\mu L,$$

$$0 < \alpha \leq \min\Big\{1,\ \frac{1}{\gamma_3\gamma_4 + L\mu\gamma_3^2,\ \frac{1}{\gamma_5 + L\mu},\ \frac{1}{\gamma_6^6}}\Big\}, \quad 0 < \beta \leq \min\{r_L,\ \alpha\}, \quad 0 < \xi \leq \beta. \tag{2.4}$$

The following lemma is crucial for the proof of the main theorem.

Lemma 2.1. ([27]) *If $\mathbf{x}_n \in \mathbf{B}(\mathbf{x}, r_L)$, then the following assertions hold:*

$$\|A_n - F'(\mathbf{x}_n)\| \leq \eta(\frac{L}{2} + \|F'(\mathbf{x}^*)\|)\|\mathbf{x}_n - \mathbf{x}^*\|$$

and A_n is invertible and $\|A_n^{-1}\| \leq \mu$.

Note that in the multi-step Ulm-Chebyshev-like method, sequence $\{B_n\}$ is generated by the algorithm except for B_0. Below, we prove that if B_0 approximates A_0^{-1}, then the sequence $\{\mathbf{x}_n\}$ generated by the multi-step Ulm-Chebyshev-like method converges locally to $\mathbf{x}^*$ with R-convergence rate 4. For this end, let B_0 satisfy that

$$\|I - B_0A_0\| \leq \xi, \tag{2.5}$$

where ξ is defined in (2.4).

Theorem 2.1. *Suppose that the Jacobian matrix $F'(\mathbf{x}^*)$ is invertible and that F' satisfies the Lipschitz condition* (2.1) *on $\mathbf{B}(\mathbf{x}^*, r_L)$. Then there exist positive numbers β and ξ such that for any $\mathbf{x}_0 \in \mathbf{B}(\mathbf{x}^*, \beta)$ and B_0 satisfying* (2.5)*, the sequence $\{\mathbf{x}_n\}$ generated by the multi-step Ulm-Chebyshev-like method with initial point $\mathbf{x}_0$ converges to $\mathbf{x}^*$. Moreover, the following estimates hold for each $n = 0, 1, \ldots$.*

$$\|\mathbf{x}_n - \mathbf{x}^*\| \leq \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^n} \tag{2.6}$$

and

$$\|I - B_n A_n\| \leq \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^n}, \tag{2.7}$$

where α and β are defined in (2.4).

Proof. We proceed by mathematical induction. Clearly, (2.6) is trivial for $n = 0$ by the assumption. By (2.4) and (2.5), we obtain

$$\alpha \leq 1 \quad \text{and} \quad \|I - B_0 A_0\| \leq \xi \leq \beta.$$

That is, (2.7) holds for $n = 0$. Now we assume that (2.6) and (2.7) hold for $n = m$. Then one has

$$\|\mathbf{x}_m - \mathbf{x}^*\| \leq \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} \tag{2.8}$$

and

$$\|I - B_m A_m\| \leq \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m}. \tag{2.9}$$

By (2.4), we get

$$\|\mathbf{x}_m - \mathbf{x}^*\| \leq \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} < \beta < r_L.$$

It follows from (2.4), (2.8), and Lemma 2.1 that

$$\begin{aligned} \|A_m - F'(\mathbf{x}_m)\| &\leq \eta\Big(\frac{L}{2} + \|F'(\mathbf{x}^*)\|\Big)\|\mathbf{x}_m - \mathbf{x}^*\| \\ &\leq \eta\Big(\frac{L}{2} + \|F'(\mathbf{x}^*)\|\Big)\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &:= \gamma_1 \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} \end{aligned} \tag{2.10}$$

and

$$\|A_m^{-1}\| \leq \mu.$$

Then

$$\begin{aligned} \|B_m\| &\leq \|B_m A_m\|\|A_m^{-1}\| \\ &\leq (1 + \|I - B_m A_m\|)\|A_m^{-1}\| \\ &\leq \mu\Big[1 + \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m}\Big] \leq 2\mu \end{aligned} \tag{2.11}$$

and

$$\begin{aligned} \|I - B_m F'(\mathbf{x}_m)\| &\leq \|I - B_m A_m\| + \|B_m\|\|A_m - F'(\mathbf{x}_m)\| \\ &\leq \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} + 2\mu\gamma_1\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &\leq (1 + 2\mu\gamma_1)\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &:= \gamma_2\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m}. \end{aligned} \tag{2.12}$$

By (1.5), we have

$$\begin{aligned}\mathbf{y}_m - \mathbf{x}^* &= \mathbf{x}_m - \mathbf{x}^* - B_m(F(\mathbf{x}_m) - F(\mathbf{x}^*)) \\ &= \mathbf{x}_m - \mathbf{x}^* - \int_0^1 B_m F'(\mathbf{x}_m^\theta)(\mathbf{x}_m - \mathbf{x}^*)d\theta \\ &= \int_0^1 [I - B_m F'(\mathbf{x}_m) + B_m(F'(\mathbf{x}_m) - F'(\mathbf{x}_m^\theta))](\mathbf{x}_m - \mathbf{x}^*)d\theta,\end{aligned}$$

where

$$\mathbf{x}_m^\theta = \mathbf{x}^* + \theta(\mathbf{x}_m - \mathbf{x}^*)$$

for $0 \le \theta \le 1$. Since

$$\|\mathbf{x}_m - \mathbf{x}^*\| \le r_L$$

and

$$\|\mathbf{x}_m^\theta - \mathbf{x}^*\| = \theta\|\mathbf{x}_m - \mathbf{x}^*\| \le \|\mathbf{x}_m - \mathbf{x}^*\| \le r_L,$$

it follows from (2.4), (2.8), (2.11), (2.12), and the Lipschitz condition that

$$\begin{aligned}\|\mathbf{y}_m - \mathbf{x}^*\| &\le \int_0^1 (\|I - B_m F'(\mathbf{x}_m)\| + L(1-\theta)\|B_m\|\|\mathbf{x}_m - \mathbf{x}^*\|)\|\mathbf{x}_m - \mathbf{x}^*\|d\theta \\ &= \|I - B_m F'(\mathbf{x}_m)\|\|\mathbf{x}_m - \mathbf{x}^*\| + \frac{L}{2}\|B_m\|\|\mathbf{x}_m - \mathbf{x}^*\|^2 \\ &\le \gamma_2\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m}\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} + L\mu\Big(\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m}\Big)^2 \\ &= \Big(\gamma_2 + L\mu\Big)\alpha^2\Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m} \\ &:= \gamma_3\alpha^2\Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m}, \end{aligned} \tag{2.13}$$

which together with (2.4), (2.8), and $\alpha \le 1$ gives

$$\begin{aligned}\|\mathbf{x}_m - \mathbf{y}_m\| &\le \|\mathbf{x}_m - \mathbf{x}^*\| + \|\mathbf{y}_m - \mathbf{x}^*\| \\ &\le \alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} + \gamma_3\alpha^2\Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m} \\ &\le (1 + \gamma_3)\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m}. \end{aligned} \tag{2.14}$$

It follows from (2.11), (2.12), and the Lipschitz condition that

$$\begin{aligned}\|I - B_m F'(\mathbf{y}_m)\| &\le \|I - B_m F'(\mathbf{x}_m)\| + \|B_m\|\|F'(\mathbf{x}_m) - F'(\mathbf{y}_m)\| \\ &\le \gamma_2\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} + 2\mu L(1 + \gamma_3)\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &\le (\gamma_2 + 2\mu L + 2\mu L\gamma_3)\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &:= \gamma_4\alpha\Big(\frac{\beta}{\alpha}\Big)^{4^m}. \end{aligned} \tag{2.15}$$

Similar to (2.13)–(2.15) and by (2.4) and $\alpha \leq 1$, we have

$$\begin{aligned}
\|\mathbf{z}_m - \mathbf{x}^*\| &\leq \|I - B_m F'(\mathbf{y}_m)\|\|\mathbf{y}_m - \mathbf{x}^*\| + \frac{L}{2}\|B_m\|\|\mathbf{y}_m - \mathbf{x}^*\|^2 \\
&\leq \gamma_4 \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} \gamma_3 \alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m} + \frac{L}{2} \times 2\mu\gamma_3^2 \alpha^4 \Big(\frac{\beta}{\alpha}\Big)^{4\times 4^m} \\
&\leq \Big(\gamma_3\gamma_4 + L\mu\gamma_3^2\Big)\alpha^3 \Big(\frac{\beta}{\alpha}\Big)^{3\times 4^m} \\
&\leq \alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{3\times 4^m}, \qquad (2.16) \\
\|\mathbf{y}_m - \mathbf{z}_m\| &\leq \|\mathbf{y}_m - \mathbf{x}^*\| + \|\mathbf{z}_m - \mathbf{x}^*\| \\
&\leq \gamma_3 \alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m} + \alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{3\times 4^m} \\
&\leq (1 + \gamma_3)\alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m} \qquad (2.17)
\end{aligned}$$

and

$$\begin{aligned}
\|I - B_m F'(\mathbf{z}_m)\| &\leq \|I - B_m F'(\mathbf{y}_m)\| + \|B_m\|\|F'(\mathbf{y}_m) - F'(\mathbf{z}_m)\| \\
&\leq \gamma_4 \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} + 2\mu L(1 + \gamma_3)\alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m} \\
&\leq (\gamma_4 + 2\mu L + 2\mu L\gamma_3)\alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} \\
&:= \gamma_5 \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m}. \qquad (2.18)
\end{aligned}$$

By (2.4), (2.16), (2.18), and $\alpha \leq 1$, we get

$$\begin{aligned}
\|\mathbf{x}_{m+1} - \mathbf{x}^*\| &\leq \|I - B_m F'(\mathbf{z}_m)\|\|\mathbf{z}_m - \mathbf{x}^*\| + \frac{L}{2}\|B_m\|\|\mathbf{z}_m - \mathbf{x}^*\|^2 \\
&\leq \gamma_5 \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} \alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{3\times 4^m} + \frac{L}{2} \times 2\mu\alpha^4 \Big(\frac{\beta}{\alpha}\Big)^{6\times 4^m} \\
&\leq (\gamma_5 + L\mu)\alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{4^{m+1}} \\
&\leq \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^{m+1}}. \qquad (2.19)
\end{aligned}$$

Consequently, (2.6) holds for $n = m + 1$, and by (2.4), (2.8), and (2.19), we get

$$\begin{aligned}
\|\mathbf{x}_{m+1} - \mathbf{x}_m\| &\leq \|\mathbf{x}_{m+1} - \mathbf{x}^*\| + \|\mathbf{x}_m - \mathbf{x}^*\| \\
&\leq \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^{m+1}} + \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} \\
&\leq 2\alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m}. \qquad (2.20)
\end{aligned}$$

By (2.4), we obtain

$$\|\mathbf{x}_{m+1} - \mathbf{x}^*\| \leq \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^{m+1}} < \beta < r_L,$$

and it follows from (2.4), (2.8), and Lemma 2.1 that

$$\|A_{m+1} - F'(\mathbf{x}_{m+1})\| \leq \gamma_1 \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^{m+1}}. \tag{2.21}$$

Together with (2.1), (2.4), (2.10), and (2.20), we have

$$\begin{aligned}\|A_{m+1} - A_m\| &\leq \|A_{m+1} - F'(\mathbf{x}_{m+1})\| + \|F'(\mathbf{x}_{m+1}) - F'(\mathbf{x}_m)\| + \|A_m - F'(\mathbf{x}_m)\| \\ &\leq \gamma_1 \alpha \Big(\frac{\beta}{\alpha}\Big)^{3^{m+1}} + 2L\alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} + \gamma_1 \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &\leq (2\gamma_1 + 2L)\alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m}, \end{aligned} \tag{2.22}$$

which follows from (2.9) and (2.11), and we get

$$\begin{aligned}\|I - B_m A_{m+1}\| &\leq \|I - B_m A_m\| + \|B_m\|\|A_{m+1} - A_m\| \\ &\leq \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} + 2\mu(2\gamma_1 + 2L)\alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &\leq (1 + 4\mu\gamma_1 + 4\mu L)\alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m} \\ &:= \gamma_6 \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^m}. \end{aligned} \tag{2.23}$$

From the fourth equation in (1.5), we obtain

$$I - \bar{B}_m A_{m+1} = (I - B_m A_{m+1})^2,$$

which together with (2.23) gives

$$\begin{aligned}\|I - \bar{B}_m A_{m+1}\| &\leq \|I - B_m A_{m+1}\|^2 \\ &\leq \gamma_6^2 \alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{2\times 4^m}. \end{aligned} \tag{2.24}$$

Notice that

$$B_{m+1} = \bar{B}_m + \bar{B}_m(2I - A_{m+1})\bar{B}_m)(I - A_{m+1}\bar{B}_m),$$

and we have

$$\begin{aligned} I - B_{m+1}A_{m+1} &= I - (\bar{B}_m + \bar{B}_m(2I - A_{m+1})\bar{B}_m)(I - A_{m+1}\bar{B}_m))A_{m+1} \\ &= (I - \bar{B}_m A_{m+1})^3. \end{aligned}$$

It follows from (2.4), (2.24), and $\alpha \leq 1$ that

$$\begin{aligned}\|I - B_{m+1}A_{m+1}\| &\leq \|I - \bar{B}_m A_{m+1}\|^2 \leq \gamma_6^6 \alpha^6 \Big(\frac{\beta}{\alpha}\Big)^{6\times 4^m} \\ &\leq \gamma_6^6 \alpha^2 \Big(\frac{\beta}{\alpha}\Big)^{4^{m+1}} \\ &\leq \alpha \Big(\frac{\beta}{\alpha}\Big)^{4^{m+1}}. \end{aligned} \tag{2.25}$$

This confirms that (2.7) holds for $n = m + 1$ and the proof is complete. □

Remark 2.1. *Under the conditions as in Theorem* 2.2*, the sequence* $\mathbf{x}^k$ *converges to the limit* $\mathbf{x}^*$ *with R-convergence rate* 4*.*

3. Numerical experiments

In this section, we report the numerical performance of the multi-step Ulm-Chebyshev-like method for solving the nonlinear operator Eq (1.1). We compare the multi-step Ulm-Chebyshev-like method (Algorithm MSUCL) with the Newton-type method (Algorithm NT) in [16], the Chebyshev-like method (Algorithm CL) in [30], the Ulm-like method (Algorithm UL) in [27], and the Ulm-Chebyshev method (Algorithm UC) in [29]. All tests were carried out in MALAB 7.10 running on a PC Intel Pentium IV with a 3.0 GHz CPU.

Example 3.1. ([30]) *We consider the following system of* 3 *nonlinear equations:*

$$\begin{cases} \cos(x_2) - \sin(x_1) = 0, \\ x_3^{x_1} - \frac{1}{x_2} = 0, \\ \exp(x_1) - x_3^2 = 0. \end{cases}$$

The Jacobian is given by

$$J(\mathbf{x}) = \begin{pmatrix} -\cos(x_1) & -\sin(x_2) & 0 \\ x_3^{x_1}\ln(x_3) & \frac{1}{x_2^2} & x_3^{x_1}\frac{x_1}{x_3} \\ \exp(x_1) & 0 & -2x_3 \end{pmatrix}$$

and

$$\mathbf{x}^* = (0.90956949452004,\ 0.66122683227485,\ 1.5758341439070)^T$$

correct to 14 *decimal places in this case. We choose the starting vector*

$$\mathbf{x}^0 = (1,\ 0.5,\ 1.5)^T$$

for $n = 0, 1, \ldots, \eta_n = \frac{1}{10}$. *For all algorithms, the stopping tolerance for the iterations is* 10^{-12}.

From Table 1, we observe that the Algorithm MSMCL converges to the solution with *R*-convergence rate 4, the Algorithm UL converges quadratically, and the Algorithm NT, the Algorithm UL, and the Algorithm UC converge cubically in the root sense.

Table 1. Values of $\|\mathbf{x}^k - \mathbf{x}^*\|$ for Example 3.1.

It.	Algorithm NT	Algorithm UL	Algorithm CL	Algorithm UC	Algorithm MSUCL
0	$2.00e-1$	$2.00e-1$	$2.00e-1$	$2.00e-1$	$2.00e-1$
1	$4.26e-3$	$5.29e-2$	$7.54e-3$	$2.22e-3$	$1.18e-4$
2	$9.56e-10$	$9.28e-5$	$2.14e-10$	$1.26e-10$	$8.99e-16$
3	$3.25e-29$	$5.29e-10$	$5.27e-30$	$4.56e-30$	
4		$5.26e-21$			

Example 3.2. ([27]) *We next consider the two-point boundary value problem*

$$\begin{cases} \mathbf{x}'' + \mathbf{x}^2 = 0, \\ \mathbf{x}(0) = \mathbf{x}(1) = 0. \end{cases} \tag{3.1}$$

We divide the interval $[0, 1]$ *into* $m + 1$ *subintervals and we get*

$$h = 1/m + 1.$$

Let $d_0, d_1, \ldots, d_{m+1}$ *be the points of subdivision with*

$$0 < d_0 < d_1 < \ldots < d_{m+1} = 1.$$

An approximation for the second derivative may be chosen as

$$\begin{cases} x_i'' = \frac{x_{i-1}-2x_i+x_{i+1}}{h^2}, \\ x_0 = x_1 = 0, \end{cases} \qquad x_i = \mathbf{x}(d_i) \text{ for } i = 1, 2, \ldots, m. \tag{3.2}$$

Let the operator ϕ*:* $\mathbf{R}^m \to \mathbf{R}^m$ *be defined by*

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, \ldots, x_m^2)^T \text{ for } \mathbf{x} = (x_1, x_2, \ldots, x_m)^T \in \mathbf{R}^m.$$

To get an approximation to the solution of (3.1), we need to solve the following nonlinear equation:

$$F(\mathbf{x}) := M\mathbf{x} + h^2\phi(\mathbf{x}) = 0, \quad \mathbf{x} \in \mathbf{R}^m,$$

where

$$M = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}_{m\times m}.$$

Obviously, $\mathbf{x}^* = 0$ *is a solution of (3.2) and*

$$F'(\mathbf{x}) = M + 2h^2\text{diag}(x_1, x_2, \ldots, x_m).$$

Hence

$$F'(\mathbf{x}^*) = M.$$

Furthermore, it is easy to verify that

$$\|F'(\mathbf{x}) - F'(\mathbf{y})\| \le 2h^2\|\mathbf{x} - \mathbf{y}\| \text{ for } \mathbf{x}, \mathbf{y} \in \mathbf{R}^m,$$

where $\|\cdot\|$ *denotes the F-norm. For different choices of m and* $\mathbf{x}_0$*, the convergence performance of the algorithm is illustrated in the following tables. Here we consider the following three problem sizes:*

(a) $m = 10$ *and* $\mathbf{x}_0 = \sigma(1, 1, \ldots, 1)^T$*;*

(b) $m = 100$ *and* $\mathbf{x}_0 = \sigma(1, 1, \ldots, 1)^T$*;*

(c) $m = 1000$ *and* $\mathbf{x}_0 = \sigma(1, 1, \ldots, 1)^T$*, where* $\sigma = 0.2$ or 0.02.

For all algorithms, the stopping tolerance for the iterations is 10^{-12}.

Tables 2–4 show that the new method has higher convergence order and from Table 5, we see that the CPU time by the Algorithm MSUCL is less than the other existing methods.

Table 2. Values of $\|\mathbf{x}^k - \mathbf{x}^*\|$ for different η_n in case (a) for Example 3.2.

σ	It.	Algorithm UL		Algorithm MSUCL		Algorithm UC	Algorithm CL	Algorithm NT
		$\eta_n := \frac{1}{20}$	$\eta_n := \frac{1}{10}$	$\eta_n := \frac{1}{20}$	$\eta_n := \frac{1}{10}$			
0.2	0	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$
	1	$1.26e-2$	$1.26e-2$	$5.42e-4$	$4.44e-4$	$4.43e-4$	$4.26e-4$	$4.26e-4$
	2	$2.96e-5$	$3.1e-5$	$5.46e-16$	$5.45e-16$	$6.15e-12$	$7.11e-12$	$3.98e-12$
	3	$2.67e-10$	$2.68e-10$			$2.33e-35$	$3.99e-35$	$4.32e-35$
	4	$3.00e-20$	$3.21e-20$					
0.02	0	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$
	1	$1.21e-4$	$1.21e-4$	$5.13e-9$	$5.14e-9$	$5.68e-7$	$5.69e-7$	$5.67e-7$
	2	$2.58e-9$	$2.62e-9$	$3.42e-42$	$3.41e-42$	$3.47e-22$	$3.48e-22$	$3.45e-22$
	3	$1.96e-18$	$2.24e-18$					

Table 3. Values of $\|\mathbf{x}^k - \mathbf{x}^*\|$ for different η_n in case (b) for Example 3.2.

σ	It.	Algorithm UL		Algorithm MSUCL		Algorithm UC	Algorithm CL	Algorithm NT
		$\eta_n := \frac{1}{20}$	$\eta_n := \frac{1}{10}$	$\eta_n := \frac{1}{20}$	$\eta_n := \frac{1}{10}$			
0.2	0	$2.00e+0$	$2.00e+0$	$2.00e+0$	$2.00e+0$	$2.00e+0$	$2.00e+0$	$2.00e+0$
	1	$3.82e-2$	$3.82e-2$	$9.63e-3$	$9.59e-3$	$1.68e-3$	$1.68e-3$	$1.68e-3$
	2	$8.87e-5$	$8.92e-5$	$1.77e-13$	$2.11e-13$	$3.67e-11$	$3.66e-11$	$3.68e-11$
	3	$7.74e-10$	$7.81e-10$			$6.57e-35$	$6.58e-35$	$6.59e-35$
	4	$8.38e-20$	$7.32e-20$					
0.02	0	$2.00e-1$	$2.00e-1$	$2.00e-1$	$2.00e-1$	$2.00e-1$	$2.00e-1$	$2.00e-1$
	1	$4.23e-4$	$3.68e-4$	$5.36e-8$	$5.37e-8$	$5.09e-6$	$5.08e-6$	$5.10e-6$
	2	$8.32e-9$	$7.74e-9$	$4.34e-29$	$4.33e-29$	$9.99e-19$	$9.98e-19$	$9.99e-19$
	3	$4.45e-18$	$5.75e-18$					

Table 4. Values of $\|\mathbf{x}^k - \mathbf{x}^*\|$ for different η_n in case (c) for Example 3.2.

σ	It.	Algorithm UL		Algorithm MSUCL		Algorithm UC	Algorithm CL	Algorithm NT
		$\eta_n := \frac{1}{20}$	$\eta_n := \frac{1}{10}$	$\eta_n := \frac{1}{20}$	$\eta_n := \frac{1}{10}$			
0.2	0	$6.32e+0$	$6.32e+0$	$6.32e+0$	$6.32e+0$	$6.32e+0$	$6.32e+0$	$6.32e+0$
	1	$1.21e-1$	$1.20e-1$	$9.87e-3$	$9.88e-3$	$5.13e-3$	$5.12e-3$	$5.14e-3$
	2	$2.96e-4$	$2.79e-4$	$1.13e-13$	$1.25e-13$	$5.55e-11$	$5.57e-11$	$5.53e-11$
	3	$2.98e-10$	$2.45e-10$			$1.03e-34$	$1.05e-34$	$1.01e-34$
	4	$2.17e-20$	$2.13e-20$					
0.02	0	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$	$6.32e-1$
	1	$1.21e-4$	$1.21e-4$	$4.71e-9$	$4.72e-9$	$4.72e-7$	$4.71e-7$	$4.73e-7$
	2	$2.28e-9$	$2.43e-9$	$4.44e-33$	$4.45e-33$	$4.18e-20$	$4.19e-20$	$4.17e-20$
	3	$1.79e-17$	$1.81e-17$					

Table 5. Averaged CPU time in seconds of all algorithms for the ten tests for Example 3.2.

n	50	100	250	500	750	1000
Algorithm UL	0.55	2.03	7.12	17.01	40.86	118.68
Algorithm UC	0.48	1.81	6.12	16.68	32.93	112.31
Algorithm NT	0.42	1.79	6.02	15.38	30.03	101.11
Algorithm CL	0.41	1.76	5.96	14.76	29.19	98.56
Algorithm MSUCL	0.26	1.01	3.99	11.23	20.44	72.21

Remark 3.1. Various indices can be employed to compare the efficiency of iterative methods. One of the most prevalent efficiency indices is introduced as follows [31, 32]:

$$\text{CR2} = \frac{\text{OC}}{\text{CPU} + \text{IT} + \text{FE} + \text{JE}},$$

where OC, IT, FE, JE, and CPU are the order of convergence, number of iterations, number of total function evaluations, total number of Jacobian evaluations, and CPU time, respectively.

We can see from Table 6 that Algorithm MSUCL is better than the other methods for solving all test problems in criteria CR2, indicating its superior performance.

Table 6. Values of CR2 for all algorithms to solve all test problems.

Method	Example 3.1	Case (a) in Example 3.2	Case (b) in Example 3.2	Case (c) in Example 3.2
Algorithm UL	0.0221	0.0176	0.0112	0.0098
Algorithm UC	0.0482	0.0158	0.0249	0.0199
Algorithm NT	0.0398	0.0268	0.0221	0.0243
Algorithm CL	0.0491	0.0289	0.0211	0.0252
Algorithm MSUCL	0.0567	0.0342	0.0298	0.0337

3.1. Chandrasekhar H-equation

The Chandrasekhar integral equation [9] which arises from radiative transfer theory is a nonlinear integral equation which gives a full nonlinear system of equations if discretized. The Chandrasekhar integral equation is given by

$$F(P,\ c) = 0, \quad P : [0,\ 1] \to \mathbf{R},$$

with parameter c and the operator F as

$$F(P,\ c)(u) = P(u) - \Big(1 - \frac{c}{2}\int_0^1 \frac{uP(v)}{u+v}dv\Big)^{-1}. \tag{3.3}$$

If we discretize the integral Eq (3.3) using the midpoint integration rule with n grid points:

$$\int_0^1 f(t)dt = \frac{1}{n}\sum_{j=1}^{n} f(t_j), \quad t_j = (j - 0.5) * h, \quad h = \frac{1}{n}, \quad 1 \le j \le n,$$

we obtain the resulting system of nonlinear equations:

$$F_i(P,\ c) = u_i - \Big(1 - \frac{c}{2n}\sum_{j=1}^{n} \frac{t_i u_i}{t_i + t_j}\Big)^{-1}, \quad 1 \le i \le n. \tag{3.4}$$

When starting with a $(1, 1, \cdots, 1)^T$ vector, the system (3.4) has a solution for all $c \in (0,\ 1)$. The c were equally spaced with $\Delta c = 0.01$ in the interval $c \in (0,\ 1)$ and we choose $n = 100$. We note that in this case the Jacobian is a full matrix for $n = 0, 1, \ldots, \eta_n = \frac{1}{10}$. For all algorithms, the stopping tolerance for the iterations is 10^{-12}.

Let $Iter_{Total}$ denote the total number of iterations for all c considered and $Iter$ be its mean iteration number. From Table 7, we find that the new method has the least total and mean number of iterations, which has the lowest computational cost and therefore is the most efficient of the other existing methods in terms of CPU time.

Table 7. Key results for the Chandrasekhar H-equation.

Statistical data	Algorithm NT	Algorithm UL	Algorithm CL	Algorithm UC	Algorithm MSUCL
$Iter_{Total}$	346	472	339	326	245
$Iter$	3.47	4.72	3.39	3.25	2.44
$CPU\ times(s)$	5248	5894	4603	4421	3199

3.2. Inverse eigenvalue problem

We consider the following inverse eigenvalue problem (IEP): given $n + 1$ real symmetric $n \times n$ matrices $\{A_i\}_{i=0}^n$ and n real numbers

$$\lambda_1^* \geq \lambda_2^* \geq \ldots \geq \lambda_n^*,$$

find a vector

$$\mathbf{c}^* = (c_1, c_2, \ldots, c_n)^T \in \mathbb{R}^n$$

such that

$$\lambda_i(A(\mathbf{c}^*)) = \lambda_i^*, \quad i = 1, \ldots, n,$$

where

$$A(c) := A_0 + \sum_{i=1}^{n} c_i A_i$$

and $\{\lambda_j(A(\mathbf{c}))\}_{j=1}^n$ are the eigenvalues of $A(\mathbf{c})$ with

$$\lambda_1(A(\mathbf{c})) \geq \lambda_2(A(\mathbf{c})) \geq \ldots \geq \lambda_n(A(\mathbf{c})).$$

The above IEP can be represented mathematically through a set of non-linear equations:

$$\mathbf{f}(\mathbf{c}) := (\lambda_1(A(\mathbf{c})) - \lambda_1^*, \lambda_2(A(\mathbf{c})) - \lambda_2^*, \ldots, \lambda_n(A(\mathbf{c})) - \lambda_n^*)^T = \mathbf{0}. \tag{3.5}$$

To further illustrate the effectiveness of the new algorithm, we present a practical engineering application in vibrations [2, 33, 34]. We consider the vibration of a taut string with n beads. Figure 1 shows such a model for the case $n = 4$.

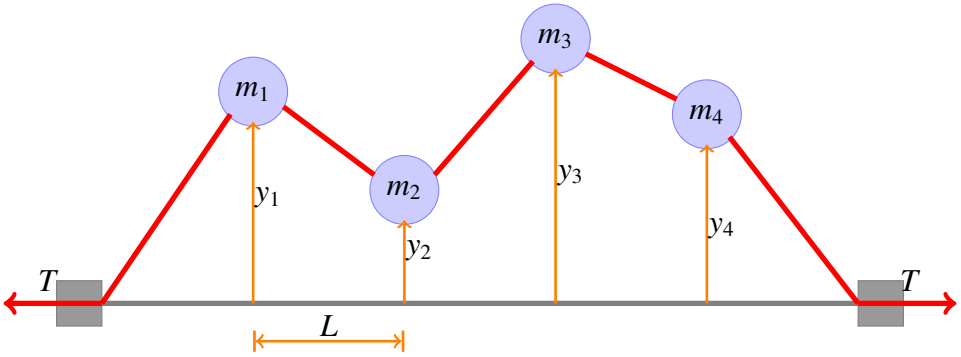

Figure 1. A string with $n = 4$ beads.

Here, we assume that the n beads are placed along the string, where the ends of the string are clamped. The mass of the jth bead is denoted by m_j. The horizontal lengths between masses m_j and m_{j+1} (and between each beads at each end and the clamped support) are set to be a constant L. The horizontal tension is set to be a constant T. Then the equation of motion is governed by

$$m_j y_j''(t) = T\frac{y_{j+1} - y_j}{L} - T\frac{y_j - y_{j-1}}{L}, \quad j = 1, \ldots, n, \tag{3.6}$$

where

$$y_0 = y_{n+1} = 0.$$

That is, the ends of the string are fixed. The matrix form of (3.6) is given by:

$$y''(t) = -CJy(t), \tag{3.7}$$

where

$$y(t) = (y_1(t), y_2(t), \ldots, y_n(t))^T, \quad C = \text{diag}(c_1, c_2, \ldots, c_n)$$

with

$$c_j = \frac{T}{m_j L},$$

and J is the discrete Laplacian matrix

$$J = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in \mathcal{S}^n.$$

The general solution of (3.7) is given in terms of the eigenvalue problem

$$CJy = \lambda y,$$

where λ is the square of the natural frequency of the vibration system and the nonzero vector y accounts for the interplay between the masses. The inverse problem for the beaded string is to compute the

masses $\{m_j\}_{j=1}^n$ so that the resulting system has a prescribed set of natural frequencies. It is easy to check that the eigenvalues of J are given by:

$$\lambda_j(J) = 4\left(\sin\frac{j\pi}{n+1}\right)^2, \quad j = 1, 2, \ldots, n.$$

Thus J is symmetric and positive definite and CJ is similar to L^TCL, where L is the Cholesky factor of

$$J = LL^T.$$

Then the inverse problem is converted into the form of the IEP where

$$A_0 = 0 \quad \text{and} \quad A_j = L^T E_j L$$

with

$$E_j = \text{diag}(e_j)$$

for $j = 1, 2, \ldots, n$. The beaded string data in Example 3.3 comes from the website: http://www.caam.rice.edu/ beads.

Example 3.3. ([2]) *This is an inverse problem for the beaded string with $n = 6$ beads, where*

$$\begin{aligned}(m_1, m_2, m_3, m_4, m_5, m_6) &= (0.017804, 0.030783, 0.017804, 0.017804, 0.030783, 0.017804)(kg),\\ (n+1)L &= 1.12395\,(meter),\\ T &= 166.0370\,(Newton),\\ \boldsymbol{\lambda}^* &= (9113.978, 30746.32, 83621.69, 133310.0, 148694.4, 193537.0)^T,\\ c^* &= (58081.57, 33592.71, 58081.57, 58081.57, 33592.71, 58081.57)^T.\end{aligned}$$

We report our numerical results for starting point

$$c^0 = (58081, 33592, 58081, 58081, 33592, 58081)^T.$$

We solve Example 3.3 using all algorithms, and the stopping tolerance is set to be

$$\|\mathbf{c}^k - \mathbf{c}^*\| \leq 10^{-12},$$

for $n = 0, 1, \ldots, \eta_n = \frac{1}{10}$. The numerical results are listed in Tables 8 and 9. We observe from Table 8 that the proposed method has higher convergence order and/or requires less operations than the other existing methods. Table 9 displays the computed masses for the beaded string. As expected, the desired masses are recovered: http://www.caam.rice.edu/ beads.

Table 8. Values of $\|\mathbf{c}^k - \mathbf{c}^*\|$ for Example 3.3.

It.	Algorithm NT	Algorithm UL	Algorithm CL	Algorithm UC	Algorithm MSUCL
0	$9.53e-1$	$9.53e-1$	$9.53e-1$	$9.53e-1$	$9.53e-1$
1	$3.45e-3$	$3.41e-2$	$3.26e-3$	$4.25e-3$	$2.11e-4$
2	$4.25e-8$	$5.24e-3$	$2.14e-8$	$8.24e-8$	$3.66e-17$
3	$2.26e-22$	$4.29e-5$	$4.36e-23$	$9.99e-23$	
4		$4.11e-9$			
5		$9.87e-17$			

Table 9. Recovered masses for Example 3.3.

	m_1	m_2	m_3	m_4	m_5	m_6
true	0.017804	0.030783	0.017804	0.017804	0.030783	0.017804
recovered	0.017804	0.030783	0.017804	0.017804	0.030783	0.017804

Example 3.4. ([1]) *This is an inverse problem with* $n = 10$. *Define*

$$A_0 = O, \quad A_1 = \frac{1}{m_1}\mathbf{e}_1\mathbf{e}_1^T, \quad A_k = (\frac{1}{\sqrt{m_1}}\mathbf{e}_1 - \frac{1}{\sqrt{m_k}}\mathbf{e}_k)(\frac{1}{\sqrt{m_1}}\mathbf{e}_1 - \frac{1}{\sqrt{m_k}}\mathbf{e}_k)^T, \quad k = 2, 3, \ldots, 6,$$

where

$$m_1 = 2, \ m_2 = m_3 = m_4 = m_5 = m_6 = 0.2.$$

Now

$$\boldsymbol{\lambda}^* = (-310.2490, -249.2218, -28.08413, 113.3087, 218.7351, 487.9554)^T.$$

Then

$$\mathbf{c}^* = (-83.47955, -53.82911, 89.13261, 40.82639, -47.78696, 21.50871)^T.$$

We report our numerical results for different starting points:

- (a) $\mathbf{c}^0 = (-77.95824, -62.08697, 96.54128, 40.10535, -44.33137, 20.79310)^T$,
- (b) $\mathbf{c}^0 = (-76.86213, -63.46336, 95.28928, 41.39452, -42.24157, 17.37889)^T$,
- (c) $\mathbf{c}^0 = (-78.58345, -65.97678, 97.83621, 43.47844, -49.26789, 23.67335)^T$,
- (d) $\mathbf{c}^0 = (-85.47863, -67.28566, 80.28746, 35.38552, -45.45096, 23.47528)^T$.

Here we take

$$B_0 = J(\mathbf{c}_0)^{-1}.$$

For all algorithms, the stopping tolerance is set to be

$$\|\mathbf{c}^k - \mathbf{c}^*\| \leq 10^{-12}.$$

Table 10 displays the error of $\|\mathbf{c}^k - \mathbf{c}^*\|$ for the above four initial points $\mathbf{c}^0$, where "It." represents the number of outer iterations, and "*" denotes that the corresponding algorithm fails to converge, respectively.

We see from Table 10 that for these choices of the initial points, Algorithm UL, Algorithm UC, and Algorithm MSUCL converge but Algorithm NT and Algorithm CL do not, and Algorithm MSUCL needs less iterations than Algorithm UL and Algorithm UC.

Table 10. Values of $\|\mathbf{c}^k - \mathbf{c}^*\|$ and It., for Example 3.4.

ini.	k	Algorithm NT	Algorithm CL	Algorithm UL $\eta_n = \frac{1}{10}$	Algorithm UC	Algorithm MSUCL $\eta_n = \frac{1}{10}$
(a)	0	$1.29e+1$	$1.29e+1$	$1.29e+1$	$1.29e+1$	$1.29e+1$
	1	$1.86e+0$	$1.86e+0$	$5.23e+0$	$4.56e+0$	$6.52e-2$
	2	$1.86e+0$	$1.85e+0$	$2.33e-2$	$6.32e-2$	$5.29e-7$
	3	$1.86e+0$	$1.86e+0$	$3.21e-3$	$9.85e-5$	$8.88e-25$
	4	$1.86e+0$	$1.86e+0$	$8.12e-5$	$5.55e-11$	
	5	$1.86e+0$	$1.86e+0$	$1.98e-8$	$6.29e-23$	
	6	$1.86e+0$	$1.86e+0$	$6.59e-11$		
	7	$1.86e+0$	$1.86e+0$	$4.22e-18$		
	It.	$*$	$*$	7	5	3
(b)	0	$1.49e+1$	$1.49e+1$	$1.49e+1$	$1.49e+1$	$1.49e+1$
	1	$1.86e+0$	$1.86e+0$	$4.26e+0$	$7.45e+0$	$1.25e-2$
	2	$1.86e+0$	$1.85e+0$	$9.84e-2$	$8.29e-2$	$9.99e-7$
	3	$1.85e+0$	$1.86e+0$	$4.25e-3$	$5.98e-5$	$3.27e-25$
	4	$1.86e+0$	$1.86e+0$	$9.86e-5$	$6.15e-11$	
	5	$1.86e+0$	$1.86e+0$	$3.26e-8$	$9.19e-23$	
	6	$1.85e+0$	$1.86e+0$	$1.29e-11$		
	7	$1.86e+0$	$1.85e+0$	$5.43e-18$		
	It.	$*$	$*$	7	5	3
(c)	0	$1.62e+1$	$1.62e+1$	$1.62e+1$	$1.62e+1$	$1.62e+1$
	1	$1.86e+0$	$1.86e+0$	$5.23e+0$	$2.11e+0$	$1.98e-2$
	2	$1.86e+0$	$1.85e+0$	$2.33e-2$	$5.37e-2$	$2.48e-7$
	3	$1.85e+0$	$1.86e+0$	$3.21e-3$	$1.11e-5$	$4.16e-25$
	4	$1.86e+0$	$1.85e+0$	$8.12e-5$	$2.51e-11$	
	5	$1.86e+0$	$1.86e+0$	$1.98e-8$	$3.26e-23$	
	6	$1.86e+0$	$1.86e+0$	$6.59e-11$		
	7	$1.86e+0$	$1.86e+0$	$4.22e-18$		
	It.	$*$	$*$	7	5	3
(d)	0	$1.74e+1$	$1.74e+1$	$1.74e+1$	$1.74e+1$	$1.74e+1$
	1	$1.86e+0$	$1.86e+0$	$4.25e+0$	$3.22e+0$	$3.21e-2$
	2	$1.86e+0$	$1.85e+0$	$5.12e-2$	$4.59e-2$	$5.29e-7$
	3	$1.86e+0$	$1.85e+0$	$1.23e-3$	$4.88e-5$	$9.99e-25$
	4	$1.86e+0$	$1.86e+0$	$5.56e-5$	$9.87e-13$	
	5	$1.85e+0$	$1.86e+0$	$8.45e-8$		
	6	$1.86e+0$	$1.86e+0$	$3.29e-13$		
	7	$1.86e+0$	$1.86e+0$			
	It.	$*$	$*$	6	4	3

4. Conclusions

In this paper, we have proposed a multi-step Ulm-Chebyshev-like method for solving nonlinear operator equations, which does not contain inverse operators in its expression, and does not require computing Jacobian matrices for solving Jacobian equations. We prove that the multi-step Ulm-Chebyshev-like method converges locally to the solution with R-convergence rate 4 under appropriate conditions. As an application, it is demonstrated how this result can be used to analyze

the Chandrasekhar integral equation and to solve the inverse eigenvalue problems. The proposed method has higher convergence order and/or requires less operations than the other existing methods, indicating its superior performance. The study of the stability analysis of our new method is also one for our future work.

Author contributions

Wei Ma: algorithms, software, numerical examples, writing–original draft, writing–review and editing; Ming Zhao: algorithms, software, numerical examples, writing–original draft, writing–review and editing; Jiaxin Li: algorithms, software, numerical examples, writing–original draft, writing–review and editing. All authors of this article have been contributed equally. All authors have read and approved the final version of the manuscript for publication.

Acknowledgments

This work was supported by the Henan Province College Student Innovation and Entrepreneurship Training Program Project (No. 202410481003).

Conflict of interest

The authors declare no conflicts of interest.

References

1. W. Ma, Two-step Ulm-Chebyshev-like Cayley transform method for inverse eigenvalue problems, *Int. J. Comput. Math.*, **99** (2022), 391–406. https://doi.org/10.1080/00207160.2021.1913728
2. W. Ma, Z. Li, Y. Zhang, A two-step Ulm-Chebyshev-like Cayley transform method for inverse eigenvalue problems with multiple eigenvalues, *AIMS Math.*, **8** (2024), 22986–23011. https://doi.org/10.3934/math.20241117
3. C. T. Wen, X. S. Chen, H. W. Sun, A two-step inexact Newton-Chebyshev-like method for inverse eigenvalue problems, *Linear Algebra Appl.*, **585** (2020), 241–262. https://doi.org/10.1016/j.laa.2019.10.004
4. Y. Wang, W. P. Shen, An extended two-step method for inverse eigenvalue problems with multiple eigenvalues, *Numer. Math. Theory Methods Appl.*, **16** (2023), 968–992. https://doi.org/10.4208/nmtma.OA-2023-0002
5. Y. S. Luo, W. P. Shen, An Ulm-like algorithm for generalized inverse eigenvalue problems, *Numer. Algorithms*, 2024. https://doi.org/10.1007/s11075-024-01845-5
6. W. Ma, Z. J. Bai, A regularized directional derivative-based Newton method for inverse singular value problems, *Inverse Probl.*, **28** (2012), 125001. https://doi.org/10.1088/0266-5611/28/12/125001
7. W. Ma, Two-step Ulm-Chebyshev-like method for inverse singular value problems, *Numer. Linear Algebra Appl.*, **29** (2022), e2440. https://doi.org/10.1002/nla.2440

8. W. Ma, X. S. Chen, Two-step inexact Newton-type method for inverse singular value problems, *Numer. Algorithms*, **84** (2020), 847–870. https://doi.org/10.1007/s11075-019-00783-x
9. C. T. Kelley, Solution of the Chandrasekhar H-equation by Newton's method, *J. Math. Phys.*, **21** (1980), 1625–1628. https://doi.org/10.1063/1.524647
10. X. Yan, X. Qian, H. Zhang, S. Song, Numerical approximation to nonlinear delay-differential Calgebraic equations with proportional delay using block boundary value methods, *J. Comput. Appl. Math.*, **404** (2022), 113867. https://doi.org/10.1016/j.cam.2021.113867
11. S. Long, Y. Zhang, S. Zhong, New results on the stability and stabilization for singular neutral systems with time delay, *Appl. Math. Comput.*, **473** (2024), 128643. https://doi.org/10.1016/j.amc.2024.128643
12. B. Morini, Convergence behaviour of inexact Newton methods, *Math. Comp.*, **68** (1999), 1605–1613. https://doi.org/10.1090/S0025-5718-99-01135-7
13. J. A. Ezquerro, M. A. Hernández, Generalized differentiability conditions for Newton's method, *IMA J. Numer. Anal.*, **22** (2002), 187–205. https://doi.org/10.1093/imanum/22.2.187
14. C. Chun, Iterative methods improving Newton's method by the decomposition method, *Comput. Math. Appl.*, **50** (2005), 1559–1568. https://doi.org/10.1016/j.camwa.2005.08.022
15. M. Frontini, E. Sormani, Some variants of Newton's method with third-order convergence, *Appl. Math. Comput.*, **140** (2003), 419–426. https://doi.org/10.1016/S0096-3003(02)00238-2
16. H. H. H. Homeier, On Newton-type methods with cubic convergence, *J. Comput. Appl. Math.*, **176** (2005), 425–432. https://doi.org/10.1016/j.cam.2004.07.027
17. M. T. Darvishi, A. Barati, A third-order Newton-type method to solve systems of nonlinear equations, *Appl. Math. Comput.*, **187** (2007), 630–635. https://doi.org/10.1016/j.amc.2006.08.080
18. J. Moser, Stable and random motions in dynamical systems with special emphasis on celestial mechanics, In: H. W. Lectures, *Annals of mathematics studies*, Princeton University Press, 1973.
19. S. Ulm, On iterative methods with successive approximation of the inverse operator, *Izv. Akad. Nauk Est. SSR.*, **16** (1967), 403–411.
20. O. H. Hald, On a Newton-Moser type method, *Numer. Math.*, **23** (1975), 411–426. https://doi.org/10.1007/BF01437039
21. H. Petzeltova, Remark on a Newton-Moser type method, *Comment. Math. Univ. Carolin.*, **21** (1980), 719–725.
22. J. M. Gutirrez, M. A. Hernández, N. Romero, A note on a modification of Moser's method, *J. Complexity*, **24** (2008), 185–197. https://doi.org/10.1016/j.jco.2007.04.003
23. A. Galperin, Z. Waksman, Ulm's method under regular smoothness, *Numer. Funct. Anal. Optim.*, **19** (1998), 285–307.
24. J. A. Ezquerro, M. A. Hernández, The Ulm method under mild differentiability conditions, *Numer. Math.*, **109** (2008), 193–207. https://doi.org/10.1007/s00211-008-0144-z
25. I. K. Argyros, On Ulm's method using divided differences of order one, *Numer. Algorithms*, **52** (2009), 295–320. https://doi.org/10.1007/s11075-009-9274-3

26. I. K. Argyros, On Ulm's method for Fréchet differentiable operators, *J. Appl. Math. Comput.*, **31** (2009), 97–111. https://doi.org/10.1007/s12190-008-0194-5

27. W. P. Shen, T. T. Wei, L. H. Peng, An Ulm-like method for solving nonlinear operator equations, *J. Nonlinear Convex Anal.*, **16** (2015), 1439–1447.

28. W. P. Shen, T. T. Wei, S. Guu, Convergence of the Ulm-like method under the Hölder condition, *J. Nonlinear Convex Anal.*, **17** (2016), 701–710.

29. J. A. Ezquerro, M. A. Hernández, An Ulm-type method with *R*-order of convergence three, *Nonlinear Anal.*, **13** (2012), 14–26. https://doi.org/10.1016/j.nonrwa.2011.07.039

30. D. K. R. Babajee, M. Z. Dauhooa, M. T. Darvishi, A. Karami, A. Barati, Analysis of two Chebyshev-like third order methods free from second derivatives for solving systems of nonlinear equations, *J. Comput. Appl. Math.*, **233** (2010), 2002–2012. https://doi.org/10.1016/j.cam.2009.09.035

31. R. H. Al-Obaidi, M. T. Darvishi, A comparative study on qualification criteria of nonlinear solvers with introducing some new ones, *J. Math.*, **2022** (2022), 4327913. https://doi.org/10.1155/2022/4327913

32. R. Erfanifar, M. Hajarian, A new multi-step method for solving nonlinear systems with high efficiency indices, *Numer. Algorithms*, **97** (2024), 959–984. https://doi.org/10.1007/s11075-023-01735-2

33. M. T. Chu, Inverse eigenvalue problems, *SIAM Rev.*, **40** (1998), 3984. https://doi.org/10.1137/S0036144596303984

34. M. T. Chu, G. H. Golub, Structured inverse eigenvalue problems, *Acta Numer.*, **11** (2002), 1–71. https://doi.org/10.1017/S0962492902000016

AIMS Press