AIMS Mathematics, 10(6): 14055–14070.
DOI: 10.3934/math.2025632
Received: 12 February 2025
Revised: 16 April 2025
Accepted: 07 May 2025
Published: 18 June 2025

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

Research article

A quartically fast iteration solver with convergence analysis for numerically determining the sign of a matrix

Ying Liu[1], Runqi Xue[2], Tao Liu[3], Shuai Wang[4] and Stanford Shateyi[5,*]

[1] Foundation Department, Changchun Guanghua University, Changchun 130033, China
[2] School of Mathematics, Sichuan University, Chengdu 610041, China
[3] School of Mathematics and Statistics, Northeastern University at Qinhuangdao, Qinhuangdao 066004, China
[4] Foundation Department, Changchun Guanghua University, Changchun 130033, China
[5] Department of Mathematics and Applied Mathematics, School of Mathematical and Natural Sciences, University of Venda, P. Bag X5050, Thohoyandou 0950, South Africa

* **Correspondence:** Email: stanford.shateyi@univen.ac.za.

Abstract: The calculation of the matrix sign function (MSF) is pivotal in numerous mathematical contexts, offering a matrix-based transformation that identifies the sign for every eigenvalue within an invertible matrix. This paper introduces a new iteration procedure tailored to effectively compute the MSF, with a particular focus on expanding the order of convergence. Our proposed solver achieves fourth-order convergence, rendering it effective for a broad spectrum of matrices. Numerical experiments for both real and complex matrices are included to support the derivations.

Keywords: matrix sign; convergence; initial matrix; numerical scheme; Newton's solver
Mathematics Subject Classification: 41A25, 65F60

1. Introduction

The matrix signum map, often called the MSF or the sign of a matrix, is a procedure that processes matrices, yielding a matrix of the similar dimension, where every element signifies the sign of the associated element (eigenvalue) in the original matrix. This notion is an extension of the sign map for scalars that works on real scalars [1]. It assigns -1 to negative scalars, +1 to positive scalars, and 0 to exact zero. The computation of the MSF is of importance in various scientific and engineering disciplines. This function is crucial for understanding the properties of matrices and their applications [2–4].

The transition from the scalar to the MSF was a natural progression, furnished to advance the investigation for the theory of matrices and to devise novel procedures in resolving equations for

matrices. The MSF first appeared in mathematical works during the late 1960s; see e.g., [5]. When an invertible matrix $Y \in \mathbb{C}^{n\times n}$ is available, then the MSF could be defined as

$$\text{sign}(Y) = \Theta, \tag{1.1}$$

where Θ is expressed as

$$\Theta = \frac{2}{\pi} Y \int_0^\infty (t^2 I + Y^2)^{-1}\, dt, \tag{1.2}$$

with I representing the unit matrix. The definition (1.2) is based on the analytical representation of the MSF using contour integration in the complex plane. This formula is particularly advantageous when the matrix M has eigenvalues clustered near the axis of imaginary. The Cauchy integral approach is sometimes also beneficial when high precision is required, as it avoids the accumulation of rounding errors that can occur in iterative methods. To compute the Cauchy integral for a matrix, numerical quadrature techniques are typically employed. The integral is approximated by discretizing the contour Γ into a finite number of points z_k and applying a quadrature rule, such as the trapezoidal rule or Gaussian quadrature. The discretized form of the integral becomes:

$$\text{sign}(Y) \approx \frac{1}{\pi i} \sum_{k=1}^{N} w_k (z_k I - Y)^{-1},$$

where w_k are the weights associated with the quadrature rule, and z_k are the quadrature points on the contour Γ [6]. Each term $(z_k I - Y)^{-1}$ involves solving a linear system, which can be performed using direct solvers or iteration schemes, depending on the size and structure of Y. For large matrices, Krylov subspace methods or preconditioning techniques can be used to accelerate the resolution of these linear systems. The choice of contour Γ is critical for the accuracy and efficiency of the method. Common choices include circles, ellipses, or parabolas that are tailored to the eigenvalue distribution of Y. Adaptive contour selection strategies can further enhance the performance by optimizing the number and placement of quadrature points. Additionally, the utilization of parallel computing procedures can sometimes reduce the computational cost, as the evaluations of $(z_k I - Y)^{-1}$ at different quadrature points are independent and can be computed concurrently.

While less commonly discussed, the MSF may find application in including stochastic differential equations [7, 8] as well as in graph theory. For instance, it could be used to analyze properties of graphs represented by adjacency matrices, such as bipartiteness or other structural characteristics based on eigenvalues. The MSF serves as a foundational element in iterative methods for computing more complex matrix functions. Enhancing the efficiency of its computation can lead to faster and more effective algorithms, which is advantageous in computational tasks requiring high performance. Recent research [9] has concentrated on enhancing algorithms for the efficient computation of the MSF, developing fast iteration solvers, and studying its relationships with other functions of matrices.

Computing (1.1) using numerical iterative methods is a compelling approach, primarily derived via tackling the nonlinear matrix equation below:

$$F(V) := V^2 - I = 0. \tag{1.3}$$

The matrix $V = \Theta$ in (1.1) is the resolution to (1.3) because it reads

$$\Theta^2 = I.$$

This work focuses on presenting a novel scheme for (1.3) by first considering the scalarized version and subsequently generalizing it to the matrix environments to ensure computational efficiency compared to existing iterative solvers for determining (1.1) for a non-singular matrix.

Newton-type iterative methods are indispensable for computing the MSF due to their efficiency, robustness, and adaptability to various matrix structures [10]. However, direct computation of the MSF is often computationally expensive, especially for large-scale matrices. Iterative methods, particularly Newton-type schemes, provide a practical alternative by approximating the solution through successive refinements. These methods are particularly advantageous because they leverage matrix-vector operations, which are highly optimized in modern computational frameworks, and they avoid the need for explicit matrix inversions, which can be numerically unstable. Driven by the practical necessity of efficiently computing the MDF, this work is motivated to the development of a higher-order iterative method. The proposed scheme is designed to maintain a computational complexity comparable to that of the optimal fourth-order methods widely reported in the existing literature. However, it distinguishes itself by exhibiting substantially larger basins of attraction, which, in turn, result in a faster convergence rate. This enhancement not only improves the reliability of the method for a broader class of initial approximations but also contributes to its overall computational efficiency and robustness in practical applications.

The rest of this research is outlined as follows:

- Section 2 reviews significant iteration methods to determine the MSF.
- Section 3 discusses the advantages of higher-order methods and furnishes a scheme for scalar nonlinear equations, which is then extended to the matrix case. A comprehensive analysis demonstrates the solver's fourth-order convergence, global applicability, and expanded attraction basins compared to similar methods, along with an examination of its stability.
- Section 4 furnishes computational results validating the analytical findings, highlighting the method's utility.
- Ultimately, Section 5 provides a conclusion to this work.

2. The concept of iterative solvers

In their seminal work [11], Kenney and Laub introduced an important and versatile family of iteration methods for computing (1.1) through the application of Padé approximations. The foundation of these methods lies in the function:

$$f(\gamma) = \frac{1}{\sqrt{1-\gamma}},$$

where the (ι_1, ι_2)-Padé approximation to $f(\gamma)$ is expressed as: $\frac{P_{\iota_1}(\gamma)}{Q_{\iota_2}(\gamma)}$. Here, P and Q are polynomials of appropriate degrees in the Padé estimation, with the condition $\iota_1 + \iota_2 \geq 1$. As discussed in [12], the iterative scheme:

$$h_{q+1} = \frac{h_q P_{\iota_1}(1-h_q^2)}{Q_{\iota_2}(1-h_q^2)} := \psi_{2\iota_1+1,2\iota_2}, \tag{2.1}$$

exhibits convergence to ± 1 with a convergence rate of $\iota_1 + \iota_2 + 1$. From this framework, the well-known 2nd-order Newton's scheme (NM) can be derived as:

$$V_{q+1} = \frac{1}{2}\left(V_q^{-1} + V_q\right), \qquad q = 0, 1, \ldots, \tag{2.2}$$

where the initial matrix is furnished as:

$$V_0 = Y, \tag{2.3}$$

with Y representing the input matrix. The significance of Newton's iterative scheme in computing (1.1) stems from its efficacy in approximating solutions iteratively. As highlighted in [13], the method boasts second-order convergence features, particularly when the starting value is in proximity to the wanted resolution. However, its performance can be limited by the need for a sufficiently accurate initial guess and its sensitivity to ill-conditioned matrices. This underscores the need for more advanced iterative methods that can overcome these limitations while maintaining computational efficiency.

Building upon (2.1), several renowned iterative techniques can be derived, including the Newton–Schulz method without the computation of the inverse with local convergence:

$$V_{q+1} = \frac{1}{2}V_q(3I - V_q^2), \tag{2.4}$$

and Halley's solver:

$$V_{q+1} = [V_q^2 + I][V_q(3I + V_q^2)]^{-1}. \tag{2.5}$$

These methods are discussed in [14]. Higher-order iterative methods, such as those based on Padé approximations or Halley's method, address these challenges by achieving faster convergence rates and greater numerical stability. These methods incorporate higher-order derivatives or additional terms in their update rules, allowing them to capture more information about the matrix and its eigenvalues. Furthermore, higher-order schemes often exhibit larger basins of attraction, meaning they are less sensitive to the choice of initial guess and can handle a wider range of matrices. This makes them suitable for applications in robust control, where the system matrices may have complex eigenvalue distributions, or in large-scale scientific computing, where efficiency and reliability are paramount. By combining the strengths of Newton-like schemes with the enhanced convergence properties of higher-order schemes, these iterative techniques provide a powerful tool for accurately and efficiently computing the MSF in diverse applications.

A fourth-order method investigated in [15] is expressed as:

$$\begin{aligned} V_{q+1} = V_q \Big((1 - 6\varsigma)I + 2(-7 + 2\varsigma)V_q^2 + (-3 + 2\varsigma)V_q^4\Big) \\ \times \Big[(1 - 2\varsigma)I - 2(3 + 2\varsigma)V_q^2 + (-11 + 6\varsigma)V_q^4\Big]^{-1}, \end{aligned} \tag{2.6}$$

where ς is a real scalar. In addition, two other 4th-order convergent solvers with global convergence properties can be derived from (2.1) as follows [11]:

$$V_{q+1} = [4V_q(I + V_q^2)][I + 6V_q^2 + V_q^4]^{-1}, \quad \text{Reciprocal of Padé [1,2]}, \tag{2.7}$$

$$V_{q+1} = [V_q^4 + 6V_q^2 + I][4V_q(I + V_q^2)]^{-1}, \quad \text{Padé [1,2]}. \tag{2.8}$$

The choice between high-order Newton-type iterative methods and direct decomposition techniques, such as Jordan or Schur decompositions, for computing the MSF depends on several factors, including the size and structure of the matrix, the desired accuracy, and the available computational resources. High-order Newton-type methods, such as Halley's method or quartic convergence schemes, are particularly advantageous when dealing with large-scale matrices or matrices with specific structures,

such as sparsity or symmetry. These methods are iterative and rely on matrix operations, which are optimized in modern computational frameworks, making them scalable and efficient for high-dimensional problems. Additionally, high-order methods are well-suited for applications requiring high precision, as they achieve faster convergence rates (e.g., cubic or quartic) compared to direct methods, which typically have a fixed computational cost. For example, in control theory or numerical analysis, where the MSF is used to solve algebraic Riccati equations or analyze system stability, high-order iterative methods are often preferred due to their ability to handle ill-conditioned matrices and their robustness to variations in the input matrix.

On the other hand, direct decomposition methods, such as Jordan or Schur decompositions, are more suitable for small - to medium-sized matrices or when an exact solution is required. These methods decompose the matrix into its canonical forms, from which the MSF can be computed directly. For instance, the Schur decomposition $M = QTQ^*$, wherein Q is unitary and T is upper triangular, allows the MSF to be computed as $\text{sign}(M) = Q \cdot \text{sign}(T) \cdot Q^*$. This approach is particularly useful when the matrix has a well-defined structure or when the decomposition itself is needed for other computations. However, direct methods are computationally expensive for large matrices, as they typically require $O(n^3)$ operations, where n is the matrix size. They are also less flexible in handling sparse or structured matrices compared to iterative methods. Therefore, direct decompositions are often reserved for problems where the matrix size is manageable, and the additional computational cost is justified by the need for exact results or the availability of precomputed decompositions.

3. A higher-order solver

Let us now focus on the scalar form of (1.3), namely,

$$f(h) = h^2 - 1 = 0.$$

Herein, $f(h)$ represents the scalar counterpart of (1.3), with solutions $h = \pm 1$. In this work, uppercase letters (for instance, V) denote matrices, while lowercase letters (e.g., h) represent scalar quantities; see also [16]. We introduce a refined three-step adaptation of Newton's method, outlined as follows:

$$\begin{cases} d_q = h_q - \frac{f(h_q)}{f'(h_q)}, & q \geq 0, \\ z_q = h_q - \dfrac{33f(h_q) - 34f(d_q)}{33f(h_q) - 67f(d_q)} \dfrac{f(h_q)}{f'(h_q)}, \\ h_{q+1} = z_q - \frac{f(z_q)}{f[z_q, h_q]}, \end{cases} \tag{3.1}$$

where $f[z_q, h_q] = \dfrac{f(z_q) - f(h_q)}{z_q - h_q}$ denotes a divided difference operator. The 2nd substep in (3.1) provides an advancement on [17]. The weights in this substep are derived using a scheme of undetermined coefficients, involving severe calculations.

Theorem 3.1. *Let $\rho \in D$ be a simple solution of the adequately smooth map $f : D \subseteq \mathbb{C} \to \mathbb{C}$. Given an initial guess h_0 sufficiently close to ρ, the scheme (3.1) tends to ρ with a convergence order of four.*

Proof. The proof involves a detailed utilization of the Taylor expansions for every substep of the iteration solver process around the zero ρ. We can obtain that (3.1) reads the following error equation:

$$\gamma_{q+1} \sim \left(-31^{-1}c_2^3\right)\gamma_q^4, \tag{3.2}$$

where $c_j = \frac{f^{(j)}(\rho)}{j!f'(\rho)}$, $\sim$ is the sign for the error equation without the truncation error term, and $\gamma_q = h_q - \rho$. It ends the proof by revealing the fourth convergence order for the error equation. □

The recurrence procedure described in (3.1) is now applicable for solving (1.3). Following this method results in

$$V_{q+1} = 4V_q\left(21I + 41V_q^2 + 4V_q^4\right)\left[17I + 166V_q^2 + 81V_q^4\right]^{-1}, \tag{3.3}$$

having the starting matrix (2.3). Analogously, the inverse formulation associated with Eq (3.3) is derived using the following structured procedure

$$V_{q+1} = \left(17I + 166V_q^2 + 81V_q^4\right)\left[4V_q\left(21I + 41V_q^2 + 4V_q^4\right)\right]^{-1}. \tag{3.4}$$

It is recalled that high-order iteration methods often deliver superior accuracy in approximating the MSF [18]. This is advantageous when working with matrices that possess small eigenvalues or demand higher-precision calculations. This feature proves especially beneficial when dealing with matrices exhibiting complex eigenvalue distributions or ill-conditioned properties, ensuring reliable and precise results in challenging scenarios. Additionally, these methods are versatile, capable of efficiently handling matrices of varying sizes and types. They can be tailored to accommodate particular matrix structures, such as sparse or symmetric matrices, making them applicable across a wide range of domains.

Theorem 3.2. *Let V_0 is an appropriate initial approximation, and suppose that Y is an invertible matrix. Given these conditions, the iteration process defined by (3.4) (or equivalently (3.3)) tends to Θ with a quartic convergence rate.*

Proof. Let A denote a nonsingular but not necessarily unique transformation matrix. The Jordan canonical form can then be expressed in the following manner:

$$J_p = A^{-1}YA = diag(J_{m_1}, J_{m_1}, \cdots, J_{m_p}). \tag{3.5}$$

Now consider a matrix Y with distinct eigenvalues $\sigma(Y) = \{\eta_1, \eta_2, \ldots, \eta_k\}$, where the sum of their multiplicities, $m_1 + m_2 + \ldots + m_p = n$. For the map f that owns $m_j - 1$ degrees of smoothness at each η_j for $j = 1, 2, \ldots, k$, as discussed in [19], the function of the matrix $f(Y)$ could be furnished as follows:

$$f(Y) = \sum_{j=1}^{k} P_j f(\eta_j) P_j,$$

where P_j is the projection matrix corresponding to η_j and

$$f(Y) = Af(J)A^{-1}, \tag{3.6}$$

wherein

$$f(J) = diag(f(J_{m_1}(\eta_1)), \cdots, f(J_{m_k}(\eta_k))), \tag{3.7}$$

with $f(J_{m_j}(\eta_j))) = \sum_{\upsilon=0}^{m_j-1} \frac{1}{\upsilon!} f^{(\upsilon)}(\eta_j) \cdot S_{m_j}^{\upsilon}$. Taking into account that m_j shows the size of the jth Jordan block corresponding to the eigenvalue η_j, it follows that we have the relation

$$J_{m_j}(\eta_j) = \begin{pmatrix} \eta_j & 1 & 0 & \cdots & 0 \\ 0 & \eta_j & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & \eta_j \end{pmatrix} =: S_{m_j} + \eta_j I. \tag{3.8}$$

In the next step, the matrix Y is decomposed using a nonsingular matrix A of the same dimensions, and the Jordan block matrix J is employed for this purpose. This results in the following decomposition:

$$Y = AJA^{-1}. \tag{3.9}$$

By employing this decomposition and carefully examining the structure of the method, an iterative procedure for the eigenvalues is derived, progressing from iteration q to iteration $q+1$ in the following manner:

$$\begin{aligned}\eta_{q+1}^{i} = &\left(17 + 166{\eta_q^i}^2 + 81{\eta_q^i}^4\right)\\ &\times\left[4\eta_q^i\left(21 + 41{\eta_q^i}^2 + 4{\eta_q^i}^4\right)\right]^{-1}, \quad 1 \le i \le n,\end{aligned} \tag{3.10}$$

where

$$n_i = \operatorname{sign}(\eta_q^i) = \pm 1. \tag{3.11}$$

In a more generalized perspective, and after applying specific simplifications, the iterative procedure described by (3.10) demonstrates that the eigenvalues progressively approach the values $n_i = \pm 1$. Specifically, this convergence can be expressed as follows:

$$\lim_{q\to\infty}\left|\frac{\eta_{q+1}^i - n_i}{\eta_{q+1}^i + n_i}\right| = 0. \tag{3.12}$$

Equation (3.12) illustrates the convergence behavior of the eigenvalues towards ± 1. As iterations progress, the eigenvalues increasingly cluster around ± 1. Following an analysis of the method's theoretical convergence, our attention now shifts to investigating the convergence rate. To achieve this, we consider the following:

$$\Lambda_q = 4V_q\left(21I + 41V_q^2 + 4V_q^4\right). \tag{3.13}$$

By employing Eq (3.13) and acknowledging that V_q constitutes rational maps of Y, while also establishing its commutativity with Θ in a manner analogous to that of Y, it becomes possible to express the following:

$$\begin{aligned}V_{q+1} - \Theta &= \left(17I + 166V_q^2 + 81V_q^4\right)\Lambda_q^{-1} - \Theta\\ &= [17I + 166V_q^2 + 81V_q^4 - \Theta\Lambda_q]\Lambda_q^{-1}\\ &= [17I + 166V_q^2 + 81V_q^4 - 4V_q(21\Theta + 41V_q^2\Theta + 4V_q^4\Theta]\Lambda_q^{-1}\\ &= [-17(V_q - \Theta)^4 + 16V_q\Theta(V^4 - 4V_q^3\Theta\\ &\quad +6V_q^2\Theta^2 - 4V_q\Theta^3 + I)]\Lambda_q^{-1}\\ &= [-17(V_q - \Theta)^4 + 16V_q\Theta(V_q - \Theta)^4]\Lambda_q^{-1}\\ &= \left(V_q - \Theta\right)^4[-17I + 16V_q]\Lambda_q^{-1}.\end{aligned} \tag{3.14}$$

Using (3.14) and 2-norm, it is possible to obtain

$$\|V_{q+1} - \Theta\| \le \left(\|\Lambda_q^{-1}\|\|17V_q - 16I\|\right)\|V_q - \Theta\|^4. \tag{3.15}$$

This result highlights that the iterative process achieves a fourth-order convergence, provided that an appropriately chosen initial matrix, such as the one given in (2.3), is employed as the starting point. The type of convergence is strong, and it is based on suitable norms. □

Attraction basins play a significant role in elucidating the global convergence characteristics of iterative schemes used to compute (1.1) [20, 21]. In the realm of iteration schemes, these basins refer to regions within the input space wherein the iteration procedure tends towards a particular resolution or exhibits a specific behavior. In the design of a solver aimed at calculating (1.1), it is imperative to guarantee that the scheme consistently converges to the desired resolution, without considering the starting approximation.

In this study, the methodologies outlined in (3.3) and (3.4) have been provided with the explicit objective of expanding the attraction regions corresponding to these iterative schemes when applied to solving the equation $f(h) = h^2 - 1 = 0$. To gain deeper insights, we proceed by examining the global convergence feature of the presented new solvers and assessing their enhanced convergence radii. This analysis is conducted by visualizing their respective attraction regions within the domain:

$$[-3, 3] \times [-3, 3],$$

while solving $f(h) = 0$.

To accomplish this, the complex plane is discretized into a structured grid of nodal points, where each point is individually assessed by considering it as an initial guess. This evaluation determines whether the iterative method tends to the solution or diverges. In cases where convergence is achieved, the corresponding points are shaded according to the number of iterations required for convergence, with the criterion being met when $|f(h_q)| \leq 10^{-3}$.

Figures 1–3 furnish the basins of attraction for various solvers. It is important to recall that the fractal nature of iteration procedures dictates their local and global convergence behavior under specific conditions. The findings indicate that the convergence radii corresponding to the methodologies outlined in (3.3) and (3.4) are significantly larger in comparison to their same-order counterparts derived by (2.1). Notably, the appearance of lighter regions in the visual representation indicates an expansion of the convergence radii, extending further to encompass solutions to (1.1).

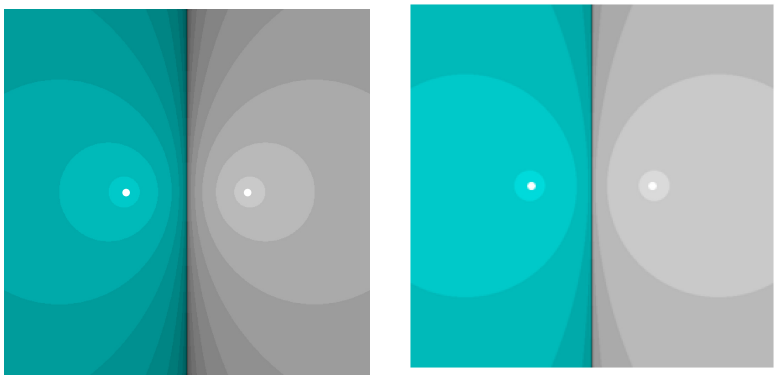

Figure 1. Convergence basins for (2.2) in left and Padé [1,2] in right.

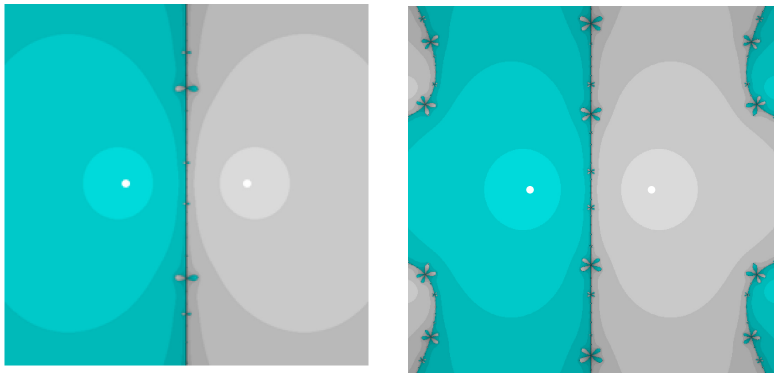

Figure 2. Convergence basins for Padé [1,3] in left and Padé [1,4] in right.

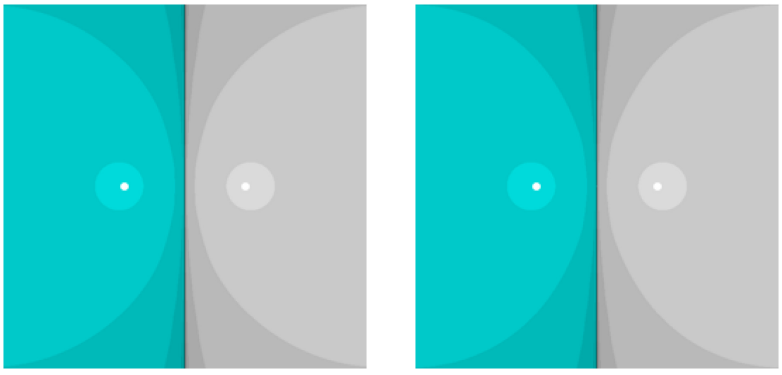

Figure 3. Convergence basins for (3.3) in left and (3.4) in right.

Theorem 3.3. *Assume Y is a nonsingular matrix. According to (3.4), the iterates $\{V_q\}_{q=0}^{\infty}$ with the initial condition $V_0 = Y$ are stable in the asymptotic sense.*

Proof. To begin, take into consideration a perturbed computation of U_q at the q^{th} iteration within the context of the computational method, as detailed further in [22]. The following equation is introduced

for every iteration cycle:

$$\widetilde{V}_q = V_q + U_q. \tag{3.16}$$

Now we consider $(U_q)^i \approx 0$ for $i \geq 2$, which is justified when doing a 1st-order error analysis, under the assumption that U_q remains small. From this, we derive the following expression for $\widetilde{V}_{q+1}$:

$$\begin{aligned} \widetilde{V}_{q+1} = &\left[11I + 106\widetilde{V}_q^2 + 51\widetilde{V}_q^4\right] \\ &\times \left[2\widetilde{V}_q\left(27I + 52\widetilde{V}_q^2 + 5\widetilde{V}_q^4\right)\right]^{-1}. \end{aligned} \tag{3.17}$$

Under the phase of converging, we assume that

$$V_q \approx \text{sign}(Y) = \Theta, \tag{3.18}$$

where we make use of established properties (with reference to the invertible matrix V and any general matrix E under consideration) [23, page 188]:

$$(E + V)^{-1} \simeq V^{-1} - V^{-1}EV^{-1}. \tag{3.19}$$

Additionally, we utilize the fact that $\Theta^{-1} = \Theta$ and $\Theta^2 = I$, which are particular cases of the identities $\Theta^{2j} = I$ and $\Theta^{2j+1} = \Theta$ for $j \geq 1$. Applying these identities, we obtain the following approximation:

$$\widetilde{V}_{q+1} \approx \left(\Theta + \frac{1}{2}U_q - \frac{1}{2}\Theta U_q \Theta\right). \tag{3.20}$$

Further simplification, combined with the expression $U_{q+1} = \widetilde{V}_{q+1} - V_{q+1}$, yields the following relation for U_{q+1}:

$$U_{q+1} \approx \frac{1}{2}U_q - \frac{1}{2}\Theta U_q \Theta. \tag{3.21}$$

From this, we conclude that the iteration sequence defined by Eq (3.4) at the $(q+1)^{\text{th}}$ iteration remains bounded. Consequently, we have the following inequality:

$$\|U_{q+1}\| \leq \frac{1}{2}\|U_0 - \Theta U_0 \Theta\|. \tag{3.22}$$

Thus, the sequence $\{V_q\}_{q=0}^{\infty}$ generated by the method in Eq (3.3) is stable. □

4. Computational study

Here, we test in Mathematica 13.3 a series of evaluations for the performance of the presented iterative solvers across a variety of problem types. The entirety of the implementations has been executed using Mathematica (refer to [24]). Various computational factors, such as identifying convergence, are systematically addressed. For clarity, the testing process is divided into two distinct categories similar to what has already been pursued in [20]: tests involving theoretical values and those pertaining to practical applications. The following iterative methods are included in the comparison:

(1) (2.2), labeled as NM;
(2) (2.5), referred to as HM3;

(3) (2.8), designated as PAM;
(4) (3.3), shown by PM1;
(5) (3.4), furnished by PM2.
(6) Additionally, the results are compared against the method introduced in [17] (denoted as UM4), defined as follows:

$$V_{q+1} = \left(5I + 42V_q^2 + 17V_q^4\right)\left[V_q\left(23I + 38V_q^2 + 3V_q^4\right)\right]^{-1}. \tag{4.1}$$

In all the iteration Newton-type schemes under comparison, the initiation value V_0 is selected as specified in (2.3). The error for each iteration is computed using the following expression:

$$E_{q+1} = \|V_{q+1}^2 - I\|_2 \leq \vartheta, \tag{4.2}$$

where ϑ denotes the predetermined stopping criterion. In terms of computational complexity, PM1 and PM2 require the same amount of calculational effort as UM4, having four matrix-by-matrix products and one matrix inversion per cycle to achieve fourth order. However, it will be observed that PM1 and PM2 outperform other solvers in terms of final elapsed time to reach the convergence phase.

It is worth emphasizing that the execution times reported for each method correspond to a single complete run of the program. These times account for all computational processes, having the evaluation of norm residuals and any other auxiliary operations necessary for the implementation.

Example 4.1. *A total of 12 random real matrices are generated by utilizing* `SeedRandom[12]`. *Subsequently, their MSFs are calculated and analyzed for comparison purposes. The random matrices are constructed within the interval* $[-10, 10]$ *and span dimensions ranging from* 100×100 *to* 1200×1200. *The computations are performed under the convergence threshold* $\vartheta = 10^{-4}$.

Tables 1 and 2 present the numerical findings associated with Example 4.1, offering compelling evidence of the effectiveness of the methodologies proposed in this work. The PM1 approach boosts calculational efficiency by minimizing the overall elapsed time required for the computation of MSF. This advancement is evident in the substantial reduction in the average CPU time, expressed in seconds, evaluated over 12 randomly generated matrices of varying dimensions. The reduction in computational time further underscores the practicality and robustness of these approaches.

Table 1. A comparison of performance, evaluated in terms of number of iterates, is provided for Example 4.1. These comparisons highlight the efficiency of the proposed methods under the specified conditions.

Matrix	$n \times n$	NM	HM3	PAM	PM1	PM2	UM4
#1	100×100	14	9	7	6	6	6
#2	200×200	16	10	8	7	7	7
#3	300×300	14	9	7	6	6	6
#4	400×400	16	10	8	7	7	7
#5	500×500	17	11	9	8	8	7
#6	600×600	22	14	11	10	10	10
#7	700×700	17	11	9	7	8	8
#8	800×800	17	11	9	8	8	8
#9	900×900	18	11	9	8	8	8
#10	1000×1000	21	14	11	9	10	9
#11	1100×1100	20	13	10	9	9	9
#12	1200×1200	19	12	10	9	8	9
Mean		17.58	11.25	9.00	**7.83**	7.91	7.83

Table 2. A comparison of performance, evaluated in terms of CPU execution times (measured in seconds), is provided for Example 4.1. These comparisons highlight the efficiency of the proposed methods under the specified conditions.

Matrix	$n \times n$	NM	HM3	PAM	PM1	PM2	UM4
#1	100×100	0.01	0.01	0.01	0.009	0.009	0.01
#2	200×200	0.05	0.04	0.03	0.03	0.03	0.05
#3	300×300	0.12	0.10	0.09	0.08	0.09	0.10
#4	400×400	0.28	0.23	0.22	0.21	0.23	0.22
#5	500×500	0.46	0.42	0.41	0.39	0.42	0.37
#6	600×600	0.92	0.78	0.77	0.76	0.78	0.76
#7	700×700	1.00	0.93	0.85	0.76	0.89	0.86
#8	800×800	1.41	1.32	1.19	1.16	1.21	1.16
#9	900×900	1.99	1.73	1.60	1.52	1.61	1.57
#10	1000×1000	3.04	2.86	2.48	2.26	2.58	2.32
#11	1100×1100	3.67	3.36	2.89	2.85	2.94	2.97
#12	1200×1200	4.52	4.00	3.76	3.61	3.31	3.66
Mean		1.46	1.31	1.19	**1.14**	1.18	1.17

Example 4.2. *In this numerical experiment, the MSF is evaluated for eight complex random matrices. The computations are performed under the convergence criterion* $\vartheta = 10^{-5}$*, as demonstrated in the Mathematica code provided below* ($I = \sqrt{-1}$)

```
SeedRandom[123];
no = 8;
Table[Y[n] = RandomComplex[{-10 - 10 I,
```

```
10 + 10 I}, {100 n, 100 n}];, {n, no}];
```

Tables 3 and 4 present detailed numerical results for Example 4.2, offering further evidence of the effectiveness of the presented solver in computing the MSF for eight randomly generated complex matrices. Similar computational tests carried out on a variety of related tests consistently validate these observations, with the PM1 method demonstrating superior performance in terms of efficiency and reliability compared to its counterparts.

Table 3. A comparison of performance, evaluated in terms of number of iterates, is provided for Example 4.2. These comparisons highlight the efficiency of the proposed methods under the specified conditions.

Matrix	$n \times n$	NM	HM3	PAM	PM1	PM2	UM4
#1	100×100	18	11	9	8	8	8
#2	200×200	20	13	10	9	9	9
#3	300×300	18	11	9	8	8	8
#4	400×400	19	12	10	9	9	9
#5	500×500	21	13	11	9	9	9
#6	600×600	19	12	10	8	8	9
#7	700×700	19	12	10	9	9	9
#8	800×800	22	14	11	10	10	10
Mean		19.500	12.250	10.000	**8.750**	8.750	8.875

Table 4. A comparison of performance, evaluated in terms of CPU execution times (measured in seconds), is provided for Example 4.2. These comparisons highlight the efficiency of the proposed methods under the specified conditions.

Matrix	$n \times n$	NM	HM3	PAM	PM1	PM2	UM4
#1	100×100	0.047	0.030	0.029	0.026	0.034	0.033
#2	200×200	0.152	0.148	0.118	0.133	0.135	0.127
#3	300×300	0.351	0.326	0.315	0.302	0.320	0.325
#4	400×400	0.715	0.650	0.670	0.686	0.711	0.678
#5	500×500	1.407	1.237	1.263	1.132	1.203	1.131
#6	600×600	2.018	1.862	1.797	1.523	1.620	1.752
#7	700×700	2.989	2.675	2.662	2.527	2.652	2.635
#8	800×800	5.010	4.577	4.115	4.069	4.191	4.188
Mean		1.586	1.438	1.371	**1.300**	1.358	1.359

5. Conclusions

The computation of the MSF presents challenges, including computational complexity and numerical stability. Addressing these challenges is crucial for developing efficient algorithms capable of handling large matrices with high accuracy, which is essential in practical applications. The MSF is a fundamental tool in linear algebra and matrix theory, with wide-ranging applications in control theory, numerical analysis, differential equations, and beyond. High-order Newton-type methods are preferred for

large-scale, sparse, or structured matrices, especially in applications requiring high precision and computational efficiency.

This paper has introduced an iterative procedure designed to effectively compute the MSF, with an emphasis on enhancing the order of convergence. The proposed method has achieved fourth-order convergence, making it suitable for a wide range of invertible matrices. In Section 3, we have highlighted the advantages of higher-order methods and have presented a scheme to solve nonlinear scalar equations, which has been subsequently extended to matrix cases. The discussion has confirmed that the solver exhibits fourth-rate convergence and has demonstrated its global applicability. Furthermore, the method has expanded the attraction basins compared to existing approaches, ensuring robust performance across diverse scenarios. Additionally, we have analyzed the stability of the method, reinforcing its reliability for practical applications. These features have collectively established the proposed method as an effective and reliable tool for computing the MSF with superior performance characteristics. Future research can focus on extending the proposed iterative method to compute other matrix functions, such as the matrix square root and matrix exponential, while preserving higher-order convergence. Additionally, investigating the performance of the method in parallel and distributed computing environments could enhance its scalability for large-scale problems. Further studies could also explore the generalization of possibilities to non-Hilbert spaces [25–27].

Author contributions

Ying Liu: Conceptualization, software; Runqi Xue: Writing-original draft, formal analysis; Tao Liu: Supervision, methodology, editing; Shuai Wang: Software, review and editing; and Stanford Shateyi; Formal analysis, software. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This research was funded by the Scientific Research Project of Jilin Provincial Department of Education (JJKH20251638KJ), the Open Fund Project of Marine Ecological Restoration and Smart Ocean Engineering Research Center of Hebei Province (HBMESO2321), the Technical Service Project of Eighth Geological Brigade of Hebei Bureau of Geology and Mineral Resources Exploration (KJ2022-021), the Technical Service Project of Hebei Baodi Construction Engineering Co., Ltd. (KJ2024-012), the Natural Science Foundation of Hebei Province of China (A2020501007), and the Fundamental Research Funds for the Central Universities (N2123015).

Conflict of interest

No conflicts of interest to disclose.

References

1. L. Hogben, *Handbook of linear algebra*, 2 Eds., New York: Chapman and Hall/CRC, 2013. https://doi.org/10.1201/b16113
2. E. D. Denman, A. N. Beavers, The matrix sign function and computations in systems, *Appl. Math. Comput.*, **2** (1976), 63–94. https://doi.org/10.1016/0096-3003(76)90020-5
3. M. Kansal, V. Sharma, P. Sharma, L. Jäntschi, A globally convergent iterative method for matrix sign function and its application for determining the eigenvalues of a matrix pencil, *Symmetry*, **16** (2024), 481. https://doi.org/10.3390/sym16040481
4. S. Wang, Z. Wang, W. Xie, Y. Qi, T. Liu, An accelerated sixth-order procedure to determine the matrix sign function computationally, *Mathematics*, **13** (2025), 1080. https://doi.org/10.3390/math13071080
5. J. D. Roberts, Linear model reduction and solution of the algebraic Riccati equation by use of the sign function, *Int. J. Control*, **32** (1980), 677–687. https://doi.org/10.1080/00207178008922881
6. N. J. Higham, *Functions of matrices: Theory and computation*, Philadelphia: Society for Industrial and Applied Mathematics, 2008. https://doi.org/10.1137/1.9780898717778
7. A. R. Soheili, M. Amini, F. Soleymani, A family of Chaplygin–type solvers for Itô stochastic differential equations, *Appl. Math. Comput.*, **340** (2019), 296–304. https://doi.org/10.1016/j.amc.2018.08.038
8. A. R. Soheili, F. Toutounian, F. Soleymani, A fast convergent numerical method for matrix sign function with application in SDEs, *J. Comput. Appl. Math.*, **282** (2015), 167–178. https://doi.org/10.1016/j.cam.2014.12.041
9. J. Golzarpoor, D. Ahmed, S. Shateyi, Constructing a matrix mid-point iterative method for matrix square roots and applications, *Mathematics*, **10** (2022), 2200. https://doi.org/10.3390/math10132200
10. J. Saak, S. W. R. Werner, Using LDL^T factorizations in Newton's method for solving general large-scale algebraic Riccati equations, *Electron. Trans. Numer. Anal.*, **62** (2024), 95–118. https://doi.org/10.1553/etna_vol62s95
11. C. S. Kenney, A. J. Laub, Rational iterative methods for the matrix sign function, *SIAM J. Matrix Anal. Appl.*, **12** (1991), 273–291. https://doi.org/10.1137/0612020
12. D. Jung, C. Chun, A general approach for improving the Padé iterations for the matrix sign function, *J. Comput. Appl. Math.*, **436** (2024), 115348. https://doi.org/10.1016/j.cam.2023.115348
13. F. Soleymani, P. S. Stanimirović, S. Shateyi, F. K. Haghani, Approximating the matrix sign function using a novel iterative method, *Abstr. Appl. Anal.*, **2014** (2014), 105301. https://doi.org/10.1155/2014/105301
14. O. Gomilko, F. Greco, K. Ziętak, A Padé family of iterations for the matrix sign function and related problems, *Numer. Linear Algebra Appl.*, **19** (2012), 585–605. https://doi.org/10.1002/nla.786
15. A. Cordero, F. Soleymani, J. R. Torregrosa, M. Zaka Ullah, Numerically stable improved Chebyshev–Halley type schemes for matrix sign function, *J. Comput. Appl. Math.*, **318** (2017), 189–198. https://doi.org/10.1016/j.cam.2016.10.025

16. J. F. Traub, *Iterative methods for the solution of equations*, New York: Prentice-Hall, 1964.

17. M. Z. Ullah, S. M. Alaslani, F. O. Mallawi, F. Ahmad, S. Shateyi, M. Asma, A fast and efficient Newton-type iterative scheme to find the sign of a matrix, *AIMS Mathematics*, **8** (2023), 19264–19274. https://doi.org/10.3934/math.2023982

18. F. Soleymani, A. Kumar, A fourth-order method for computing the sign function of a matrix with application in the Yang–Baxter-like matrix equation, *Comput. Appl. Math.*, **38** (2019), 64. https://doi.org/10.1007/s40314-019-0816-6

19. R. Bhatia, *Matrix analysis*, New York: Springer, 1997. https://doi.org/10.1007/978-1-4612-0653-8

20. L. Shi, M. Z. Ullah, H. K. Nashine, M. Alansari, S. Shateyi, An enhanced numerical iterative method for expanding the attraction basins when computing matrix signs of invertible matrices, *Fractal Fract.*, **7** (2023), 684. https://doi.org/10.3390/fractalfract7090684

21. Y. Feng, A. Z. Othman, An accelerated iterative method to find the sign of a nonsingular matrix with quartical convergence, *Iran. J. Sci.*, **47** (2023), 1359–1366. https://doi.org/10.1007/s40995-023-01506-7

22. B. Iannazzo, *Numerical solution of certain nonlinear matrix equations*, PhD Thesis, Universita Degli Studi di Pisa, Pisa, Italy, 2007.

23. G. W. Stewart, *Introduction to matrix computations*, New York: Academic Press, 1973.

24. S. Mangano, *Mathematica cookbook*, O'Reilly Media, 2010.

25. H. Brezis, *Functional analysis, Sobolev spaces and partial differential equations*, New York: Springer, 2011. https://doi.org/10.1007/978-0-387-70914-7

26. K. Yosida, *Functional analysis*, Berlin, Heidelberg: Springer, 1995. https://doi.org/10.1007/978-3-642-61859-8

27. W. Rudin, *Functional analysis*, McGraw Hill, 1991.

AIMS Press