Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. *, No. *, *, pp. 1-12
DOI:10.22034/cmde.2025.67379.3203

A rapidly convergent iteration scheme for computing the matrix sign function of invertible matrices

Malik Zaka Ullah[1,*], Abdulaziz Saeed Alqarni[1], Ali Saleh Alshomrani[1], Mohammed Almuzaini[2], and Mir Asma[3]

[1]Mathematical Modelling and Applied Computation (MMAC) Research Group, Department of Mathematics, King Abdulaziz University, Jeddah, 21589, Saudi Arabia.

[2]Department of Mathematics, Jamoum University College, Umm Al-Qura University, Makkah, 25375, Saudi Arabia.

[3]Department of Mathematics, Saveetha School of Engineering, SIMATS, Saveetha University, Chennai 602105, Tamil Nadu, India.

Abstract

This paper presents a novel 6th-rate multi-step iteration method to calculate the matrix sign function of an invertible matrix. The proposed scheme is constructed using rational approximations and carefully designed weight functions, leading to enhanced computational efficiency and high accuracy. A detailed convergence analysis is provided, including the derivation of an explicit error expression that rigorously establishes the method's sixth-order convergence rate. Additionally, we generalize the approach to matrix iteration frameworks, demonstrating that the eigenvalues of the generated iterates asymptotically converge to their theoretical limits. Numerical tests are conducted to validate the analytical findings and illustrate the superior performance of the presented method compared to existing techniques.

Keywords. Matrix function, Eigenvalues, Nonsingular, Higher order, Iteration solver.

2010 Mathematics Subject Classification. 41A25; 65F60.

1. Introductory notes

1.1. **Background.** The matrix sign function (MSF) is an eminent function in computational mathematics with theoretical and practical importance. One of its most crucial roles is in solving algebraic Riccati equations (AREs), which frequently arise in control theory and optimization [11]. The Riccati equation plays a key role in stabilizing linear systems, particularly in linear-quadratic regulator (LQR) problems, Kalman filtering, and H-infinity control. The connection between the MSF and AREs is established through a transformation where solving the Riccati equation is equivalent to deriving the invariant stable subspace of a Hamiltonian matrix. The sign function provides a robust method for this computation via separating the unstable and stable eigenvalues [3]. Compared to direct eigenvalue-based methods, sign function iterations tend to be more numerically stable and less sensitive to perturbations, making them highly suitable for large-scale systems.

Beyond control theory, the MSF has applications in computing invariant subspaces, which is essential in eigenvalue problems and model reduction techniques [4]. The ability of the sign function to classify eigenvalues based on their location in the complex plane makes it a powerful tool for spectral decomposition and numerical stability analysis [13]. It is widely used in computational physics and quantum mechanics, particularly in lattice quantum chromodynamics (QCD), where it helps evaluate the overlap Dirac operator [15]. The accurate computation of the MSF in such cases is critical because it influences the stability and convergence of iterative solvers employed in large-scale simulations.

Another application of the MSF is in the study of matrix polynomials and nonlinear matrix equations. These equations arise in areas such as systems biology, where they model complex interactions in dynamical systems [14].

Received: 18 May 2025; Accepted: 12 April 2026.
∗ Corresponding author. Email: zmalek@kau.edu.sa.

The sign function is particularly useful in solving quadratic matrix equations of the form

$$B_1X^2 + B_2X + B_3 = 0,$$

which appear in queueing theory and stochastic processes [2]. Moreover, the function is extensively used in the analysis of stability in electrical power networks and mechanical systems, where determining the spectral properties of system matrices is crucial for predicting long-term behavior. Since these systems often involve large, sparse matrices, iterative methods via the sign function furnish an efficient computational approach that avoids full matrix diagonalization.

It is a generalize of the sign function for scalars, which is expressed for a complex number ζ that does not lie on the axis of imaginary as [8]:

$$\operatorname{sign}(\zeta) = \begin{cases} -1, & \text{if } \operatorname{Real}(\zeta) < 0, \\ 1, & \text{if } \operatorname{Real}(\zeta) > 0. \end{cases}$$

This function can be generalized to matrices using various approaches, including Jordan form and polynomial interpolation techniques. Given $A \in \mathbb{C}^{n\times n}$ having no pure eigenvalues, then its MSF is well-defined. If the Jordan decomposition of A is furnished as $A = ZJZ^{-1}$, where J is a block diagonalized matrix including Jordan blocks associated to eigenvalues in the left/right half-planes. The MSF of A can be presented via [17]:

$$\operatorname{sign}(A) = Z \begin{bmatrix} -I_p & 0 \\ 0 & I_q \end{bmatrix} Z^{-1}.$$

Other formats of the MSF provide additional insights and computational advantages. One such concise expression is:

$$\operatorname{sign}(A) = A(A^2)^{-1/2},$$

wherein $B^{1/2}$ represents the principal square root for B. Another integral representation of the function is:

$$\operatorname{sign}(A) = \frac{2}{\pi} A \int_0^\infty (t^2 I + A^2)^{-1} dt.$$

Recalling that the MSF has several key properties, which are concluded in the following theorem.

Theorem 1.1. *[8] Suppose $A \in \mathbb{C}^{n\times n}$ constitute a square matrix whose eigenvalues do not possess purely imaginary components, and suppose* $\operatorname{sign}(A) = S$. *The following properties hold:*

1. *$S^2 = I$, meaning that S is an involutory matrix.*
2. *The eigenvalues of S are exclusively ± 1.*
3. *The matrices S and A commute, i.e., $SA = AS$.*
4. *If A has only real elements, then so does S.*
5. *The matrices $(S+I)/2$ and $(-S+I)/2$ act as projectors into the subspaces that remain invariant corresponds to the eigenvalues of A in the right and left half-planes, in a respective manner.*

While S is a square root for the unit matrix, it is not necessarily I or $-I$ except in the case where the spectrum of A is completely contained in one of the half-planes [18]. Furthermore, even though the eigenvalues of S are strictly ± 1, its norm can be large arbitrarily.

1.2. **Block Representation and Special Cases.** A useful block form of the MSF is derived for a 2×2 block matrix of the structure:

$$P = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix},$$

wherein $A, B \in \mathbb{C}^{n\times n}$ and neither AB nor BA has eigenvalues on the real negative axis. Here the MSF is presented by:

$$\operatorname{sign}(P) = \begin{bmatrix} 0 & V \\ V^{-1} & 0 \end{bmatrix},$$

wherein $V = A(BA)^{-1/2}$. A notable special case occurs when $B = I$, leading to:

$$\operatorname{sign}\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix} = \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix}.$$

1.3. **The Sign Decomposition and Sensitivity.** The MSF allows for a decomposition of A into two factors [8]:

$$A = SN, \quad S = \operatorname{sign}(A), \quad N = (A^2)^{1/2}.$$

This factorization is crucial in sensitivity analysis, where perturbations in A lead to changes in S. The derivative in the Fréchet sense of the sign satisfies the Sylvester equation:

$$NL + LN = \Delta A - S\Delta AS,$$

where L is the derivative in the Fréchet sense and ΔA is the perturbation.

1.4. **Computational Methods.** Several numerical techniques exist for computing $\operatorname{sign}(A)$. Using the Schur factorization $A = \Gamma\nu\Gamma^*$, where Γ is unitary and ν is upper triangular, the sign function can be computed as:

$$\operatorname{sign}(A) = \Gamma \operatorname{sign}(\nu)\Gamma^*.$$

The computation of $\operatorname{sign}(\nu)$ follows a recurrence based on its triangular structure.

Another widely used iterative approach is Newton's scheme, given by [5]:

$$Y_{l+1} = 2^{-1}(Y_l + Y_l^{-1}), \quad Y_0 = A. \tag{1.1}$$

This method is quadratically convergent and ensures rapid convergence when well-conditioned. A more general class of iterations is derived from Padé approximants, leading to the iteration [10]:

$$Y_{l+1} = Y_l \frac{p_{\ell m}(I - Y_l^2)}{q_{\ell m}(I - Y_l^2)}, \tag{1.2}$$

where $p_{\ell m}$ and $q_{\ell m}$ are polynomials selected to improve convergence.

Although significant progress has been made, there remains a pressing need for further advancements in the development of iterative methods that simultaneously achieve rapid convergence and enhanced calculational robustness [19]. Driven via such issues [20, 24], this study proposes a 6th-order iteration scheme based upon refined rational estimations combined with weighting functions. The organization for the rest of the present work is written in what follows.

- In section 2, a new strategy in resolving nonlinear scalar equations is introduced, which is then extended to the matrix framework. A rigorous analytical study establishes the 6th-rate convergence of the contributed procedure.
- Subsequently, section 3 delves into the global convergence characteristics of the presented procedure by examining its regions of attraction basins.
- Section 4 provides computational tests which validate the analytical insights and further demonstrate the efficacy of the furnished approach.
- Lastly, section 5 provides a short conclusion, highlighting the contributions of this study.

2. Deriving a new iteration solver

To compute $\operatorname{sign}(A)$ iteratively, we must solve the following:

$$G(Y) = Y^2 - I. \tag{2.1}$$

We seek a matrix Y such that:

$$G(Y) = 0 \Rightarrow Y^2 = I.$$

Toward this goal, we shift our focus to the scalar counterpart of Equation (2.1), expressed as [21]

$$g(y) = y^2 - 1 = 0.$$

In this setting, the function $g(y)$ serves as the scalar equivalent of (2.1), whose solutions are given by $y = \pm 1$. To address this problem, we introduce an improved multi-step iteration method which leverages rational approximates along with weight functions (see [12, 23] for background), formulated as follows:

$$\begin{cases} h_l = y_l - g'(y_l)^{-1} g(y_l), \qquad l = 0, 1, \ldots, \\ z_l = y_l - \dfrac{1000g(y_l) - 1001g(h_l)}{1000g(y_l) - 2001g(h_l)} (g'(y_l)^{-1} g(y_l)), \\ v_l = z_l - g[z_l, y_l]^{-1} g(z_l), \\ y_{l+1} = v_l - g[v_l, h_l]^{-1} g(v_l), \end{cases} \tag{2.2}$$

where the divided difference operator is defined by $g[a_1, a_2] := \frac{g(a_1)-g(a_2)}{a_1-a_2}$. The structure of (2.2) has been deliberately designed to fulfill two key objectives: first, to derive a novel iterative method that is distinct from the Padé family of methods, and second, to ensure computational efficiency in evaluating the MSF while exhibiting a globally convergent behavior, as will be elaborated in this study.

Theorem 2.1. *Suppose $\alpha \in D$ is a simple solution of the adequately smooth map $g : D \subseteq \mathbb{C} \to \mathbb{C}$. If the initial estimate y_0 is chosen sufficiently close to α, then the iteration process (2.2) converges to α with a sixth-order rate of accuracy.*

Proof. Since the steps of the proof are based heavily on Taylor series and would be straightforward, here we only report the final error equation. The error equation associated with (2.2) takes the form:

$$e_{l+1} = y_l - \alpha = -\frac{1}{1000} b_2^5 e_l^6 + \mathcal{O}(e_l^7), \tag{2.3}$$

where $b_i = \frac{1}{i!} \frac{g^{(i)}(\alpha)}{g'(\alpha)}$, $i \geq 2$. With this, the proof is concluded by establishing the 6th-rate convergence for the equation of error. □

The iteration procedure presented in (2.2) is now ready to be employed for solving (2.1). By following this methodology, we arrive at

$$Y_{l+1} = Y_l \left(7005I + 35005Y_l^2 + 20991Y_l^4 + 999Y_l^6\right)$$

$$\left[1001I + 21009Y_l^2 + 34995Y_l^4 + 6995Y_l^6\right]^{-1}, \tag{2.4}$$

while initializing the process with the matrix

$$Y_0 = A. \tag{2.5}$$

Analogously, the reciprocal formulation associated with Equation (2.4) can be derived by adopting the following systematic procedure:

$$Y_{l+1} = \left(1001I + 21009Y_l^2 + 34995Y_l^4 + 6995Y_l^6\right) \left[Y_l \left(7005I + 35005Y_l^2 + 20991Y_l^4 + 999Y_l^6\right)\right]^{-1}. \tag{2.6}$$

To provide better intuition for the derived iteration scheme, we note that each transformation step in (2.4) corresponds to a local linearization of the nonlinear system around the current approximation. This reformulation highlights the connection between the proposed iteration and classical Newton-type methods, with an additional correction term derived from the Padé-type expansion that improves convergence near the solution.

Theorem 2.2. *Suppose that Y_0 represents a well-chosen initial approximation and that A is an invertible matrix. Using the conditions, the procedure defined by (2.6) (also (2.4)) exhibits convergence to S, attaining a 6th-order speed.*

Proof. It is worth noting that the matrix A undergoes decomposition through the nonsingular Z of the same size, coupled with the Jordan block matrix J, thereby yielding the decomposition below:

$$A = ZJZ^{-1}. \tag{2.7}$$

Via exploiting this factorization and performing a comprehensive structural analysis of the method, a procedure is devised to determine the eigenvalues (ω). The framework progresses from the l-th iterate to the next iterate $l+1$ according to the following relation:

$$\begin{aligned}\omega_{l+1}^i &= \left(1001 + 21009{\omega_l^i}^2 + 34995{\omega_l^i}^4 + 6995{\omega_l^i}^6\right) \\ &\times \left[\omega_l^i\left(7005 + 35005{\omega_l^i}^2 + 20991{\omega_l^i}^4 + 999{\omega_l^i}^6\right)\right]^{-1}, \quad 1 \le i \le n,\end{aligned} \tag{2.8}$$

whereas

$$s_i = \operatorname{sign}(\omega_l^i) = \pm 1. \tag{2.9}$$

From an analytic perspective, and after applying the necessary simplifications, the iterative method outlined in (2.8) establishes that the eigenvalues exhibit asymptotic convergence to $s_i = \pm 1$. More rigorously, this procedure is encapsulated by the following formulation:

$$\lim_{l\to\infty}\left|\frac{\omega_{l+1}^i - s_i}{\omega_{l+1}^i + s_i}\right| = 0. \tag{2.10}$$

The relation (2.10) characterizes the asymptotic behavior of the eigenvalues, illustrating their progressive tendency to cluster around the limiting values ± 1 as the iteration procedure advances. After every successive iterate, the eigenvalues converge more precisely toward these limits, exhibiting an increasingly refined accuracy. Considering the theoretical basis for the convergence of the proposed method, we now proceed to analyze its convergence rate in detail. To facilitate this examination, we undertake the following approach:

$$H_l = Y_l\left(7005I + 35005Y_l^2 + 20991Y_l^4 + 999Y_l^6\right). \tag{2.11}$$

By employing (2.11) and acknowledging that Y_l constitutes a rational function of A, while also guaranteeing that Y_l preserves commutativity with S in the same way as A, the formulation could be derived as

$$\begin{aligned}Y_{l+1} - S &= \left(1001I + 21009Y_l^2 + 34995Y_l^4 + 6995Y_l^6\right)H_l^{-1} - S \\ &= [1001I + 21009Y_l^2 + 34995Y_l^4 + 6995Y_l^6 - SH_l]H_l^{-1} \\ &= [1001I + 21009Y_l^2 + 34995Y_l^4 + 6995Y_l^6 \\ &- Y_l[7005S + 35005Y_l^2S + 20991Y_l^4S + 999Y_l^6S]H_l^{-1} \\ &= [-1001(Y_l - S)^6 + 999Y_lS(Y_l - S)^6]H_l^{-1} \\ &= (Y_l - S)^6[-1001I + 999Y_l]H_l^{-1}.\end{aligned} \tag{2.12}$$

Applying (2.12) in conjunction with the 2-norm, we obtain

$$\|Y_{l+1} - S\| \le \left(\|H_l^{-1}\|\|999Y_l - 1001I\|\right)\|Y_l - S\|^6. \tag{2.13}$$

This result underscores that the iterative procedure achieves a sixth-order convergence rate, provided that an appropriately selected initial matrix is utilized. □

Several comments are in order:

- The iteration converges with sixth order for well-conditioned matrices, leading to rapid convergence.
- When implemented with suitable preconditioning, the method is numerically stable [22].
- Unlike methods that require eigendecomposition, this iteration works directly with matrix operations, making it efficient for large-scale problems.
- The iteration consists of matrix additions and inversions, which can be efficiently parallelized on modern computing architectures.

3. Global convergence

Drawing attraction basins when proposing iteration schemes for deriving the MSF is important for understanding the global behavior, efficiency, and stability of the method. While demonstrating global convergence, except on the imaginary axis, is an important aspect, the visualization of attraction basins serves multiple additional purposes. These plots help illustrate how different regions in the complex plane behave under iteration, revealing whether an algorithm reliably converges to the correct solution from a broad range of initial conditions. For an iterative method to be practically useful, it must exhibit a large basin of attraction, ensuring convergence from a wide variety of starting matrices. Studies have shown that Newton's scheme, as well as high-rate schemes such as Padé-based iterations, exhibit distinct attraction basin structures that can be analyzed to determine their effectiveness in approximating the MSF [9].

In addition to demonstrating global convergence, the analysis of attraction basins enables direct comparisons between various numerical methods in terms of efficacy. Some schemes converge faster in most regions, while others may require significantly more iterations in certain areas of the complex plane [7]. By visualizing attraction basins, researchers can identify whether a given iterative scheme is computationally superior to existing methods. Moreover, attraction basin plots provide insight into the rate of convergence, as regions where iterations converge rapidly to the MSF tend to be well-defined, whereas slow convergence may manifest as fractal-like boundaries or irregular regions, [16]. Another important aspect of attraction basin visualization is its role in detecting numerical instability and sensitivity to initial conditions. Some iterative methods may exhibit chaotic behavior, oscillations, or divergence for particular choices of initial matrices. The imaginary axis, in particular, represents a challenging region where traditional iterative schemes may struggle due to the ill-conditioned nature of the issue. Attraction basins help identify these problematic regions, allowing researchers to modify their iterative algorithms to enhance stability.

Finally, attraction basins provide a valuable tool for identifying exceptional points where convergence is slow or does not occur [6]. In numerical methods, it is well known that some regions in the complex plane can lead to indefinite iteration cycles or stagnation, particularly when eigenvalues of the matrix lie near to the axis of imaginary. By studying attraction basins, it becomes possible to diagnose these slow-converging or non-converging regions and improve the iteration process through scaling, preconditioning, or adaptive step-size adjustments.

In this study, the methodologies formulated in (2.4) and (2.6) have been specifically devised to expand the attraction regions corresponding to these iterative methods in the context of solving the equation $g(y) = y^2 - 1 = 0$. To achieve a deeper insight into their effectiveness, we undertake an analysis of the global convergence properties of the presented solvers, demonstrating their superior convergence radii. This is accomplished by visualizing their respective attraction basins within the domain: $[-4, 4] \times [-4, 4]$, while solving $g(y) = 0$.

To facilitate this investigation, the complex plane is discretized into a mesh of points, where each point serves as a starting point in the iterative process. The behavior of each point is then examined to obtain if the solver diverges or converges. For points exhibiting convergence, a shading scheme is employed via the quantity of iterates needed for convergence, with the stopping criterion defined as $|g(y_l)| \leq 10^{-2}$. Black areas show no convergence.

Figures 1-3 present the convergence areas for the iterative methods (2.4)-(2.6) as well as four different higher order members of the Padé family of methods. The findings indicate that both (2.4) and (2.6) exhibit extensive and globally effective convergence regions, further confirming their robustness and efficiency.

4. Results

Here, we conduct an evaluation of the contributed methods by analyzing their performance for a broad spectrum of problems. The whole computational framework has been implemented using Mathematica 14.0 [1]. A methodology has been employed to address various computing considerations, including the identification of convergence behavior.

Two of the iterative methods included in this comparative study are as follows: the method formulated in (1.1), named as NM2; the scheme provided via (4.1), designated as Halley:

$$Y_{l+1} = Y_l(I + 3Y_l^2)(3I + Y_l^2)^{-1}, \tag{4.1}$$

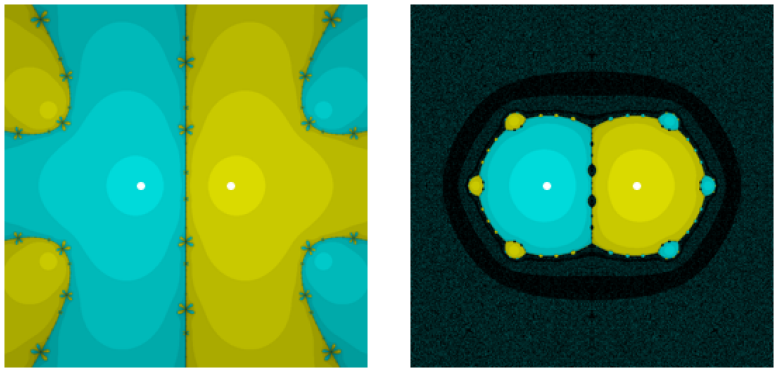

FIGURE 1. Convergence areas in Padé[1,4] in left and Padé[4,1] in right.

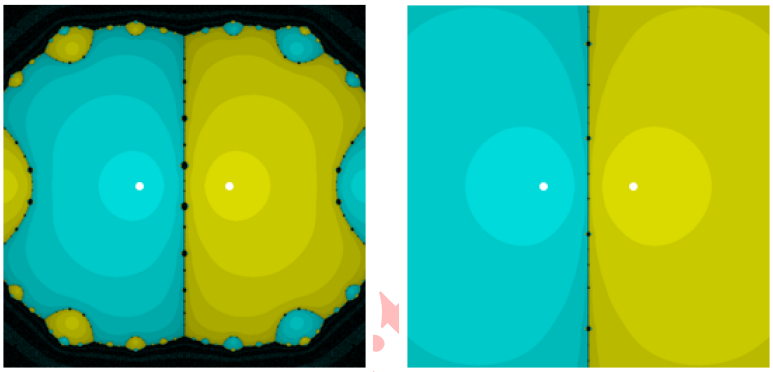

FIGURE 2. Convergence areas in Padé[2,2] in left and Padé[3,2] in right.

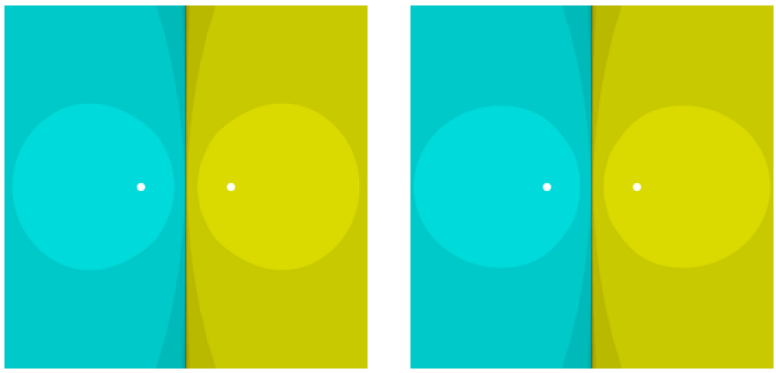

FIGURE 3. Convergence areas in (2.4) in left and (2.6) in right.

the iterative procedure presented in (2.4), denoted as Proposed1; the method described in (2.6), labeled as Proposed2; and the fourth-order approach proposed by Zaka Ullah et al., referred to as Zaka [25], which is expressed as follows:

$$Y_{l+1} = \left(5I + 42Y_l^2 + 17Y_l^4\right)\left[Y_l\left(23I + 38Y_l^2 + 3Y_l^4\right)\right]^{-1}. \tag{4.2}$$

In the Newton-like iteration methods examined in the present work, Y_0 is selected via (2.5). The error is quantified utilizing the expression below:

$$E_{l+1} = \|Y_{l+1}^2 - I\|_2 \leq \zeta, \tag{4.3}$$

wherein ζ denotes the tolerance, which serves as the stop criterion.

Example 4.1. A collection of 10 real matrices is randomly produced by utilizing the Mathematica function `SeedRandom[56789]` to ensure reproducibility. Once these matrices are constructed, their respective MSFs are calculated and analyzed to enable a detailed comparative assessment. The generated matrices are defined within the numeric domain $[-20, 20]$ and exhibit sizes spanning from 100×100 to 1000×1000. All computational procedures are performed under the convergence tolerance parameter $\zeta = 10^{-6}$.

TABLE 1. A performance comparison is carried out by analyzing the required number of iterations to achieve convergence for Example 4.1.

Size	Newton	Halley	Zaka	Proposed1	Proposed2
100×100	15	9	7	5	5
200×200	19	12	9	7	7
300×300	17	11	7	6	6
400×400	20	13	9	8	8
500×500	22	14	10	8	8
600×600	19	12	9	7	7
700×700	18	11	8	7	7
800×800	25	16	11	9	9
900×900	19	12	8	7	7
1000×1000	26	17	12	10	10
Mean	20.0	12.7	9.0	7.4	7.4

TABLE 2. A comparative analysis is performed via the CPU time for Example 4.1.

Size	Newton	Halley	Zaka	Proposed1	Proposed2
100×100	0.017	0.017	0.012	0.009	0.009
200×200	0.067	0.071	0.052	0.052	0.047
300×300	0.166	0.137	0.106	0.117	0.121
400×400	0.376	0.341	0.293	0.298	0.316
500×500	0.720	0.572	0.514	0.485	0.476
600×600	0.928	0.763	0.736	0.666	0.647
700×700	1.185	1.024	0.946	0.982	0.916
800×800	2.293	1.931	1.742	1.718	1.620
900×900	2.296	1.981	1.628	1.646	1.690
1000×1000	4.082	3.635	3.235	2.998	3.102
Mean	1.21	1.04	0.92	0.89	0.89

The numerical findings associated with Example 4.1 are presented in Tables 1-2, offering compelling validation of the effectiveness of the proposed methods. The Proposed1 method demonstrates superior calculational efficacy by minimizing the whole quantity of iterations required for MSF computation. This enhancement is evident in a significant reduction in the mean CPU time, reported in seconds, across a set of ten experiments of varying dimensions.

Example 4.2. The MSF is determined for a collection of ten complex matrices. The computation is performed under the constraint $\zeta = 10^{-6}$, ensuring rigorous accuracy in the evaluation process. The generation and implementation of these randomly selected matrices are demonstrated in the Wolfram 14.0 code snippet provided below

```
SeedRandom[56789];
numb = 10;
Table[A[n] = RandomComplex[{-20 - 20 I,
      20 + 20 I}, {100 n, 100 n}];, {n, 1, numb}];
```

Tables 3-4 present a comprehensive computational comparison for Example 4.2, thereby substantiating the effectiveness of the contributed solver to compute the MSF for a collection of 8 randomly produced complex matrices. The reliability of these findings is further reinforced through extensive numeric tests performed on a wide range of associated test cases. Notably, among the various iterative methods examined, the Proposed1 method demonstrates superior robustness and efficacy, surpassing alternative approaches in terms of calculational precision and convergence performance.

TABLE 3. A performance comparison is carried out by analyzing the required number of iterations in Example 4.2.

Size	Newton	Halley	Zaka	Proposed1	Proposed2
100×100	17	11	8	7	7
200×200	25	16	11	9	9
300×300	18	11	8	7	7
400×400	21	13	9	8	8
500×500	21	13	9	8	8
600×600	22	14	10	8	8
700×700	20	13	9	8	8
800×800	23	15	10	8	8
900×900	24	16	11	9	9
1000×1000	25	16	11	10	10
Mean	21.6	13.8	9.6	8.2	8.2

TABLE 4. A comparative analysis is performed via the CPU time, measured in seconds in Example 4.2.

Size	Newton	Halley	Zaka	Proposed1	Proposed2
100×100	0.029	0.024	0.033	0.033	0.040
200×200	0.171	0.159	0.145	0.152	0.142
300×300	0.303	0.303	0.296	0.293	0.297
400×400	0.790	0.693	0.622	0.637	0.652
500×500	1.202	1.135	1.020	1.097	1.106
600×600	2.036	1.953	1.806	1.697	1.731
700×700	2.762	2.672	2.461	2.469	2.475
800×800	4.546	4.375	3.805	3.487	3.483
900×900	6.946	6.445	5.616	5.319	5.244
1000×1000	10.478	9.159	7.798	8.010	8.176
Mean	2.92	2.69	2.36	2.31	2.33

The results presented in Tables 1-4 provide a thorough comparative analysis of the efficacy and numerical performance of the contributed iteration schemes in deriving the MSF. The discussion below highlights key observations and insights derived from these results.

4.1. **Discussions.** For Example 4.1, a collection of ten random real matrices was used to assess the convergence behavior of various iterative schemes. Table 1 clearly demonstrates that the contributed methods outperform classical approaches, such as Newton's and Halley's solvers, in terms of the quantity of iterations needed for convergence. Specifically, the Proposed1 and Proposed2 methods exhibit the lowest iteration counts, with an average of 7.4 iterations, compared to 20.0 for Newton's method and 12.7 for Halley's method. This improvement in convergence rate directly impacts computational efficiency.

Table 2 further reinforces the superiority of the proposed methods by reporting CPU execution times for each approach. The results indicate that Proposed1 and Proposed2 methods achieve the fastest computations, with an average execution time of approximately 0.89 seconds. In contrast, Newton's method requires an average of 1.21 seconds, and Halley's method takes around 1.04 seconds. This reduction in computational cost highlights the effectiveness of the contributed algorithms in handling large-scale real matrices.

Example 4.2 extends the analysis to complex matrices, with a similar numerical evaluation conducted using ten randomly generated test cases. Table 3 presents the iteration counts required for convergence. The trends observed in this case align with those for real matrices: the Proposed1 and Proposed2 methods consistently require fewer iterations (averaging 8.2) than their conventional counterparts, with Newton's method requiring an average of 21.6 iterations and Halley's method needing 13.8 iterations.

The computational time analysis in Table 4 further supports these findings. The proposed methods demonstrate superior efficiency, with an average CPU execution time of approximately 2.31 seconds for Proposed1 and 2.33 seconds for Proposed2. In contrast, Newton's method requires an average of 2.92 seconds, and Halley's method takes 2.69 seconds. This indicates that the proposed approaches maintain their efficiency advantage even when applied to complex matrices.

The numerical findings confirm that the proposed iterative methods consistently outperform existing schemes in terms of both convergence rate and computational cost. By reducing the number of iterations needed for MSF computation, these methods effectively enhance efficiency while maintaining robust accuracy. Additionally, the observed improvements remain consistent across different problem dimensions and matrix types (real and complex), demonstrating the general applicability of the proposed solvers.

Moreover, the relatively stable performance of Proposed1 and Proposed2 suggests that these methods are well-suited for large-scale computations, where reducing execution time is crucial. The improvements observed in both iteration counts and CPU time make the proposed methods strong candidates for solving high-dimensional matrix problems efficiently.

The experimental results provide strong evidence supporting the effectiveness of the proposed algorithms, making them valuable tools for practical applications in computational mathematics and scientific computing. Future work may explore further optimizations, such as parallel implementations or adaptations for specialized hardware, to further enhance computational performance.

Beyond the presented benchmark tests, the proposed iterative scheme exhibits promise for large-scale computational problems, particularly in areas such as optimal control, nonlinear inverse problems, and parameter estimation. The method's rapid convergence and reduced Jacobian evaluations render it suitable for real-time control applications where computational efficiency is critical. Future investigations will focus on adapting the proposed solver to practical engineering systems governed by differential and algebraic equations, such as those arising in nonlinear control.

5. Conclusion and future works

We have developed a new multi-step iterative procedure for computing the MSF, designed to achieve sixth-order convergence. By employing rational approximations in conjunction with weight functions, we have formulated an iterative method that diverges from the classical Padé-based approaches while maintaining computational efficiency. The proposed scheme has been rigorously analyzed, and its convergence properties have been thoroughly established.

Through a detailed theoretical investigation, we have demonstrated that the iteration process exhibits rapid convergence to the MSF when initialized with a well-chosen approximation. The presented convergence analysis has

confirmed that the method attains a sixth-order rate of accuracy, as evidenced by the derived error equation. Furthermore, we have shown that the eigenvalues of the iterates asymptotically converge to their limiting values, reinforcing the robustness of the proposed approach.

The theoretical foundation laid in this study opens new avenues for future research. In forthcoming work, we aim to explore optimized implementations of the iterative scheme tailored for large-scale and sparse matrices, leveraging efficient numerical linear algebra techniques. Additionally, we plan to investigate the extension of this method to other matrix functions, such as the principal matrix square root and sign-preserving matrix decompositions. These directions hold promising potential for further enhancing computational performance and broadening the applicability of the proposed approach in various scientific and engineering domains.

Acknowledgment

This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under grant no. (GPIP: 556-130-2024). The authors, therefore, acknowledge with thanks DSR for technical and financial support.

References

[1] M. L. Abell and J. P. Braselton, *Mathematica by Example*, Fifth Edition, Academic Press, The Netherlands, 2017.
[2] D. Bini, B. Iannazzo, and B. Meini, *Numerical Solution of Algebraic Riccati Equations*. SIAM, PA, 2012.
[3] R. Byers, *Solving the algebraic Riccati equation with the matrix sign function*, Linear Algebra Appl., *85* (1987), 267–279.
[4] E. D. Denman and A. N. Beavers Jr, *The matrix sign function and computations in systems*, Appl. Math. Comput., *2* (1976), 63–94.
[5] Y. Feng and A. Zaka Othman, *An accelerated iterative method to find the sign of a nonsingular matrix with quartical convergence*, Iran. J. Sci., *47* (2023), 1359-1366.
[6] C. Getz and J. Helmstedt, *Graphics with Mathematica Fractals, Julia Sets, Patterns and Natural Forms*, Elsevier, Amsterdam, 2004.
[7] O. Gomilko, F. Greco, and K. Ziętak, *A Padé family of iterations for the matrix sign function and related problems*, Numer. Lin. Alg. Appl., *19* (2012), 585–605.
[8] N. J. Higham, *Functions of Matrices: Theory and Computation*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2008.
[9] D. Jung and C. Chun, *A general approach for improving the Padé iterations for the matrix sign function*, J. Comput. Appl. Math., *436* (2024), Art. ID: 115348.
[10] C. S. Kenney and A. J. Laub, *Rational iterative methods for the matrix sign function*, SIAM J. Matrix Anal. Appl., *12* (1991), 273–291.
[11] P. Lancaster and L. Rodman, *Algebraic Riccati Equations*. Oxford University Press, 1995.
[12] T. Lotfi and M. Momenzadeh, *Constructing an efficient multi-step iterative scheme for nonlinear system of equations*, Comput. Methods Differ. Equ., *9* (2021), 710-721.
[13] D. S. Mackey, N. Mackey, and F. Tisseur, *Structured Tools for Structured Matrices*, Electronic Journal of Linear Algebra, *10* (2003), 106-145.
[14] T. Nadaf and T. Lotfi, *Finding the solution of a nonlinear matrix problem by an inverse-free iteration scheme*, Int. J. Math. Model. Comput., *12* (2022), 163-171.
[15] H. Neuberger, *Exactly massless quarks on the lattice*, Phys. Lett. B., *417* (1998), 141-144.
[16] L. Rani and M. Kansal, *Numerically stable iterative methods for computing matrix sign function*, Math. Meth. Appl. Sci., *46* (2023), 8596-8617.
[17] J. D. Roberts, *Linear model reduction and solution of the algebraic Riccati equation by use of the sign function*, Int. J. Cont., *32* (1980), 677–687.
[18] S. Salehi and T. Lotfi, *A globally convergent numerical method with fourth order for computing the matrix sign*, Math. Methods Appl. Sci. , *48* (2025), 7460-7468.

[19] P. Sharma and M. Kansal, *Extraction of deflating subspaces using disk function of a matrix pencil via matrix sign function with application in generalized eigenvalue problem*, J. Comput. Appl. Math., *442* (2024), 115730.
[20] P. Sharma and M. Kansal, *An iterative formulation to compute the matrix sign function and its application in evaluating the geometric mean of Hermitian positive definite matrices*, Mediterr. J. Math., *22* (2025), 1-23.
[21] F. Soleymani, P. S. Stanimirović, and I. Stojanović, *A novel iterative method for polar decomposition and matrix sign function*, Discrete Dyn. Nat. Soc., *2015* (2015), 1-11.
[22] F. Soleymani, F. K. Haghani, and S. Shateyi, *Several numerical methods for computing unitary polar factor of a matrix*, Adv. Difference Equ. , *2016* (2016), 1-11.
[23] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall: New York, NY, USA, 1964.
[24] N. Zainali and T. Lotfi, *A globally convergent variant of mid-point method for finding the matrix sign*, Comp. Appl. Math., *37* (2018), 5795-5806.
[25] M. Zaka Ullah, S. Muaysh Alaslani, F. Othman Mallawi, F. Ahmad, S. Shateyi, and M. Asma, *A fast and efficient Newton-type iterative scheme to find the sign of a matrix*, AIMS Math., *8* (2023), 19264–19274.