Theoretical Extensions and Numerical Analysis of Parameter-Manipulated HSS-Type Iterations for Sparse Linear Systems

Shaymaa Hatem Oraibi

Ministry of Transport, Administrative and Financial Department

Email: shaimaahatem911@gmail.com

Abstract

Background: Large, sparse linear systems of the form Ax = b is ubiquitous in Management Information Systems and business analytics, appearing in supply-chain optimization, PageRank and web-scale ranking, portfolio optimization, and networked stochastic models. The Hermitian/Skew-Hermitian Splitting (HSS) method is a flexible iterative framework for such problems, but its practical performance is highly sensitive to the choice of the acceleration parameter and can be slow or unstable without careful tuning.

Materials and Methods: We develop two parameter-manipulation extensions of HSS: (i) Adaptive Parameter HSS (AP-HSS), which dynamically updates the scalar acceleration parameter using a low-cost residual-minimization heuristic and safeguarded quadratic interpolation among candidate values; and (ii) Cyclic Parameter HSS (CP-HSS), which applies a pre-defined geometric cycle of parameters with the possibility of precomputing factorizations for the cycle. Both methods are compared against standard HSS and NHSS using representative test matrices drawn from synthetic convection–diffusion discretizations, Markov transition systems, and portfolio/KKT problems; prototype implementations employ sparse direct solvers and measure iteration counts, CPU time, and relative residual norms.

Results: Numerical experiments on the representative suite show that both AP-HSS and CP-HSS consistently outperform fixed-parameter HSS in iterationwise residual reduction and final residuals for convection–diffusion and PSD portfolio tests; AP-HSS delivers the most robust improvement across heterogeneous problems by adapting to local spectral characteristics, while CP-HSS is highly efficient when a small cycle of parameters can be precomputed and amortized. Stochastic/Markov examples highlight the need for additional stabilization: naive parameter choices can destabilize HSS,

whereas AP-HSS's guarded evaluation improves robustness though additional preconditioning may still be required to reach strict tolerances.

Conclusion: Structured parameter tuning for HSS-type iterations provides a useful approach for enhanced speed and robustness for linear solvers in Management Information Systems (MIS) and business analytics. AP-HSS offers a black-box, automated solution, thus minimizing computationally expensive spectral precomputation as part of an analysis pipeline. CP-HSS offers an alternative with improved performance when precomputation is not prohibitively expensive. These techniques enable efficient large-scale simulations and analytics processes to be implemented in business intelligence; future efforts should include developing formal convergence proofs, exploring broader generalizations of the methodology, predicting parameters using machine learning, and deploying these techniques onto very large scales.

Keywords: HSS iterations; adaptive parameter selection; cyclic parameter methods; sparse linear systems; Management Information Systems.

1. Introduction

The present commercial landscape is fundamentally influenced by data, which also creates, or at least accelerates, the necessity for organizations to be engaged with various types of analytics to remain a competitive force. Management Information Systems (MIS), as a discipline that involves a crosspoint of business and technology, interacts with its own set of very difficult problems, including those found in logistics (e.g. the problem of traveling salesman, etc.), recommendation systems (e.g. Amazon and others, etc.), and financial risk (e.g. evaluating a portfolio's risk or credit scoring with its associated risk), to name just a few, of which there are more. Increasingly many of the problems have the disadvantage of being described mathematically become increasingly formalized particularly in the form of mathematical models, where solutions are typically arrived at by solving a series of large systems of linear equations, in the form typically expressed as $Ax = b$. Matrices arising from these types of examples, whether they represent web graphs for PageRank [1], or transition matrices to analyze customer behavior with a transition occurring between two states through the use of Markov chains, are both extremely large, but also extremely sparse with the overwhelming majority of its entries containing zero. Indeed, the "big data" context

exists in so that the dimensionality n may reach millions or more. In that sort of context, removing themselves away from any sort of classical solutions is going to simply waste their time if it doesn't just become impossible to adequately obtain a satisfactory solution. In particular, direct solvers such as Gaussian Elimination, which change the matrix structure, become computationally expensive due to their $O(n^3)$ arithmetic complexity and very large memory required to contain fill-in elements—the non-zero values produced during the factorization process [2]. As a result, iterative methods, which generate a series of consecutive approximations that improve without changing the original matrix, have become the preferred and often the only possible computational path.

Among the many iterative solvers, the Hermitian/Skew-Hermitian Splitting (HSS) method, developed by Shokrpour and Ebadi (2022) [3], has become quite popular due to its solid theoretical underpinning. It is based on the interesting splitting of the coefficient matrix A as $H + S$, where H and S are the Hermitian and Skew-Hermitian parts of A, respectively. One of the main advantages of the HSS iteration is its unconditional convergence for a significant class of problems — when A is positive definite. Its two-step structure, based on solving systems with either $\alpha I + H$ or $\alpha I + S$, takes advantage of the properties of H and S and often ends up resulting in simpler subsystems that may be easy to solve. This robustness makes it a reliable method for a numerical analyst. Nevertheless, this potential is mitigated by a serious and widely recognized vulnerability: the convergence rate of the HSS algorithm is exceedingly sensitive to the selection of the acceleration parameter, α. The theoretical selection of the best α, which will minimize the spectral radius of the iteration matrix and therefore achieve fastest convergence, is a non-negligible challenge which may issue in expensive estimates of the extreme eigenvalues of H [4]. As noted in several of the most recent analyses, this practical responsibility often results in practitioners relying on ad hoc or suboptimal choices, which may severely compromise the method's efficiency and reliability in large-scale computational settings, which is particularly alarming in relation to financial and economic modeling applications [5-6].

This paper, hence, addresses this important gap between theoretical potential and practical performance. Our main theoretical contribution is the expansion of the HSS framework by creating an innovative iteration scheme that maybe helpful when

systematically varying parameter α. We offer two specific measures: A dynamic adaptive parameter selection (AP-HSS) forms the basis of adaptive α selection by heuristically minimizing the residual norm and a cyclic parameter (CP-HSS) dynamically utilizing a chosen sequence of α to eliminate stagnation by poorly selected parameter α. To test these theoretical expansions, we present our numerical contribution, which includes a substantive comparison of the proposed techniques versus the conventional HSS and its normalized version (NHSS) through a series of test problems. Test problems employed would consist of synthetic matrices as well as actual datasets from some of the fundamental MIS domains, such as network analysis and portfolio optimization. Finally, we provide the MIS practitioner community with a more efficient, automatic, and parameter-less sensitive solver that facilitates the computational ability and speed of conducting business analytics on a large scale. Finally, the remainder of the paper is structured in the following way: Section 2 provides a full literature review, Section 3 details the proposed methods, Sections 4 and 5 present the numerical results and discussion, and finally Section 6 offers conclusions and recommendations for future research directions.

2. Literature Review:

The solution of large, sparse linear systems forms the computational backbone of numerous scientific and engineering disciplines. The development of iterative methods began with the first methods developed at that time such as the Jacobi method, the Gauss-Seidel method, and the Successive Over-Relaxation method (SOR). Although these classical solvers, outlined in the landmark paper by Peng and Vempala (2024) [7], were easy to implement, they often had slow convergence rates or required strong diagonal dominance to guarantee convergence, making them the least efficient and unreliable options on more complex, ill-conditioned matrices that are common in modern applications. This limitation provided the background from which many advanced matrix splitting methods could rely on certain structures of the matrix for better convergence. The method of Hermitian/Skew-Hermitian Splitting (HSS) was a major achievement in this line of methods, which was developed by Shokrpour and Ebadi (2022) [3]. This procedure is based on the elegant and universal splitting of any matrix A into its Hermitian (H) and Skew-Hermitian (S) components. The HSS

iteration utilizes this splitting in a two-step structure, where the iteration alternates between solving subsystems involving the matrices $\alpha I + H$ and $\alpha I + S$. A key appealing feature of the method is that it converges unconditionally under the assumption that A is positive definite, a property which was proven in the original paper. The stability of the HSS procedure ignited a myriad of research on HSS-type methods. For example, Channark et al. (2022) created the Normalized HSS (NHSS) method, which is designed for non-Hermitian positive determinant matrices.[1] In this way, the NHSS is particularly important for many business applications where coefficient matrices are not Hermitian, but are still positive definite, like the case in modeling Markov chains. Further expanding the set of HSS-related procedures, Khorsand Zak and Abbaszadeh Shahri (2025) presented the Positive-Definite and Skew-Hermitian Splitting (PSS) method [8]; Dehghan and Shirilord (2022) later generalized this method by curating a Generalized HSS (GHSS) iteration to showcase the ongoing evolutionary emergence of this research area [9].

Nevertheless, the theoretical appeal of the HSS method is marred by an important and well-described practical issue - the convergence rate is highly sensitive to the choice of the acceleration parameter, α. From a theoretical standpoint, we note that the best parameter, α*, is the one that minimizes the spectral radius of the HSS iteration matrix, as thoroughly studied in Zhang et al. (2023) [10]. They provided elegant theoretical bounds for the optimal parameter α*, usually stated in terms of the extreme eigenvalues of the Hermitian part H; however, the bounds, as noted in Dereziński and Yang (2024) [11], suffer from severe practical restrictions, translating these to eigenvalue estimates in general for large-scale sparse matrices is an expensive computational exercise, generally at least as costly as the actual cost of solving the linear system, thereby losing the value of an iterative method. This gap between theory and practice has led to some research on adaptive schemes for parameter selection. Channark et al. (2022), for example, applied an adaptive scheme based on minimizing the residual norm, but it calls for extra matrix-vector products with a per-iteration costing more computationally [1]. Wang et al. (2024) developed a self-adaptive PSS technique [12], but it is a heuristic and not theoretically guaranteed to work across different problem classes. More recently, Xiong et al. (2025) used a statistical learning for predicting a good parametrization [13], but it increases complexity and data dependency, thereby being

unsuitable for a more general use, black-box solver. Despite the contributions here, we note a common tradeoff remains: effectiveness vs computational cost and robustness.

The need to solve these linear systems is especially urgent in the areas of Management Science and Management Information Systems (MIS), where data-driven models take center stage. For example, Ye et al. (2021) solved a large linear system when examining the stability and resilience of a global production network [14], a pivotal analysis for assessing supply chain risk. Similarly, in customer analytics, Zhou et al. (2024) solved a Markov chain system to derive the long-term state distribution modeling customer brand loyalty [15], an indispensable finding for assessing marketing strategy. Finally, Tadonki (2021) evinces this dependency in financial analytics when showing that the specialized interior-point methods for large-scale portfolio optimization [16], a classic quadratic programming problem, depend upon solving the associated Karush-Kuhn-Tucker (KKT) conditions, which are also sparse linear systems. The results of a detailed survey of computational trends in management information systems by Kar et al. (2025) demonstrate an increasing dependence on such numerical methods [17]. In summary, the HSS methods and its variations provide an empirical basis for robust analysis, and while applications in business analytics have been accepted, they are still not as pervasive as they could be. There seems to be a need or even necessity for a straightforward, effective, and sound manipulation of established parameters and analytics, which can harness the full potential of the HSS-type methods, making them not only more powerful than the existing vigilance methods in MIS, but also able to be used by the MIS practitioner community with minimal investment in expertise of numerical linear algebra.

3. Materials and Methods

3.1. Theoretical Framework and Preliminaries

Our methodological approach is based on the standard Hermitian/Skew-Hermitian Splitting (HSS) approach. For any square matrix $A \in C^{n\times n}$, this decomposition is unique and is written as:

$$A = H + S \quad (1)$$

where $H = \frac{A + A^H}{2}$ is the Hermitian component and $S = \frac{A - A^H}{2}$ is the skew-Hermitian component. Based on this basic splitting, Shokrpour and Ebadi (2022) proposed the HSS iteration scheme [3], which can be written in fixed-point matrix splitting format as:

$$x^{k+1} = M(\alpha)^{-1}N(\alpha)x^k + M(\alpha)^{-1}b \quad (2)$$

The definition of the iteration matrices is $M(\alpha) = (\alpha I + H)(\alpha I + S)$ and $N(\alpha) = (\alpha I - H)(\alpha I - S)$, with $\alpha > 0$ being the acceleration parameter cycling the iteration scheme. The convergence of this scheme depends on the spectral radius of the iteration matrix $\rho\left(M(\alpha)^{(-1)}N(\alpha)\right)$, and the necessary and sufficient condition for convergence is:

$$\rho\left(M(\alpha)^{(-1)}N(\alpha)\right) < 1 \quad (3)$$

For non-Hermitian positive definite matrices, which often appear in business applications like Markov chain modeling, Channark et al. (2022) introduced the Normalized HSS (NHSS) method. The NHSS iteration preserves the formal structure of (2) but uses altered splitting matrices that result in better convergence properties for this important class of problems [1].

The key issue driving our current work relates to the sensitive dependence of the convergence rate on the parameter α. Zhang et al. (2023) have shown through theoretical examinations that there are elegant bounds for the optimal $\alpha*$ which minimizes the spectral radius in equation (3) [10], but these often require costly eigenvalue evaluations which are not necessarily viable for large-scale challenges. This has been consistently noted in recent computations studies, such as those by Xiong et al. (2025) [13], who acknowledged the extreme expense involved in evaluating spectral radius for matrices arising from activities in modern business analytics.

Formally, our research problem may now be summarized as follows: Given the iterative intensity defined in equation (2) and the convergence criterion defined in (3), we would like to develop efficient methods for pleasingly generating a parameter sequence $\{\alpha_k\}$ to accelerate convergence with while remaining acceptable in costing in solving large-scale sparse linear systems which are prevalent for Management Information Systems.

3.2. Proposed Method 1: Adaptive Parameter HSS (AP-HSS)

The Adaptive Parameter HSS (AP-HSS) method is our main contribution to the parameter selection problem. The basic idea is to allow for updating acceleration parameter α at each iteration, depending on current convergence behavior, rather than using this parameter value consistently throughout the solution. The motivation for this approach is that the locally optimal parameter will likely change as iterations progress and the approximations to the solution improve.

Specifically, we are developing a mechanism for adaptation based on the estimation of minimizing residual reduction at each new step. In particular, we will describe the parameter update for this as follows.

$$\alpha_{k+1} = argmin_{\alpha} \left|\left|r_{(k+1)(\alpha)}\right|\right|_2 \quad (4)$$

where $r_{(k+1)(\alpha)} = b - Ax^{k+1}$ represents the residual at the $k+1$ th iteration. In order to avoid expensive computations, we use a cheap heuristic approximation from the on-dimensional minimization problem. Following the suggestion from Channark et al. (2022) [1] for related splitting methods, we approximate the residual norm reduction by leveraging a low-rank update approach that only requires matrix-vector products instead of solving an entire linear system. The complete AP-HSS algorithm is summarized as Table 1, showing the complete step-by-step pseudocode to follow for implementation.

Table 1. Pseudocode for Adaptive Parameter HSS (AP-HSS) Method

Step	Description
1	**Input:** A, b, x^0, α_0, max_iter, tol
2	**Compute:** $H = \frac{A + A^H}{2}$, $S = \frac{A - A^H}{2}$
3	**Precompute:** LU factorizations of $(\alpha_{0I} + H)$ and $(\alpha_{0I} + S)$ if feasible
4	**For** k = 0, 1, 2, … , max_iter
5	Solve $(\alpha_{kI} + H)x^{k+\frac{1}{2}} = (\alpha_{kI} - S)x^k + b$
6	Solve $(\alpha_{kI} + S)x^{k+1} = (\alpha_{kI} - H)x^{k+\frac{1}{2}} + b$
7	Compute residual $r_{k+1} = b - Ax^{k+1}$
8	**If**

Step	Description
9	Update $\alpha_{k+1} = adaptive_{parameter_{update}(\alpha_k, r_{k+1}, x^{k+1})}$
10	**End For**
11	**Output:** Solution x^{k+1}

As mentioned in step 9, the adaptive parameter update procedure uses a protected one-dimensional minimization method. We apply quadratic interpolation, which evaluates the residual norm at three candidate values of α, and chooses the one that minimizes the residual norm, as used by Wang et al. (2024) in a parameter adaptation method for positive-definite splitting methods [12]. This is a computationally efficient method for balancing choosing a good parameter, as it only introduces matrix-vector products in lieu of solving a system of linear equations.

3.3. Proposed Method 2: Cyclic Parameter HSS (CP-HSS)

The CP-HSS method offers a different method of choosing the parameters which avoids the drawbacks of complete adaptive schemes while keeping the benefits of fixed parameters. The main idea is to use a known cyclic sequence of parameters $\{\alpha_1, \alpha_2, \dots, \alpha_m\}$ that will repeat during the iteration process. The key reason for this is based on the theoretical work presented by Dereziński and Yang (2024) in terms of parameterized iterative methods [11], where different parameter values effectively target different parts of the matrix spectrum.

The justification of the cyclic process is that in effect, sampling number of different parameter values throughout the iteration gives a better average convergence, compared to the average worst-case of a single minimum value. This is particularly useful in practice, since it may be difficult to determine which fixed parameter is best.

To help build the cyclic parameter sequence, we propose a geometric progression over a reasonable range of the optimal parameter:

$$\alpha_i = \alpha_{\min} \times \left(\frac{\alpha_{max}}{\alpha_{min}}\right)^{\frac{i-1}{m-1}} for\ i = 1, 2, \dots, m \quad (5)$$

The bounds α_{min} and α_{max}n be approximated using inexpensive matrix norms, or gauged along the way through practical experience with similar problems. In recent work by Dehghan and Shirilord (2022) on generalized HSS methods, they indicated

that the geometrically distributed parameter sequences demonstrated more robustness in rate of convergence [9].

The complete CP-HSS algorithm is provided in Table 2, which describes the implementation steps for the cyclic parameter strategy.

Table 2. Pseudocode for Cyclic Parameter HSS (CP-HSS) Method

Step	Description
1	**Input:** A, b, x^(0), cycle_params = {α_1, α_2, ..., α_m}, max_iter, tol
2	**Compute:** H = (A + A^H)/2, S = (A - A^H)/2
3	**Precompute:** LU factorizations for each unique (α_iI + H) and (α_iI + S) in cycle
4	**For** k = 0, 1, 2, ..., max_iter
5	j = mod(k, m) + 1 // Cycle index
6	α_current = cycle_params[j]
7	Solve (α_currentI + H)x^(k+1/2) = (α_currentI - S)x^(k) + b
8	Solve (α_currentI + S)x^(k+1) = (α_currentI - H)x^(k+1/2) + b
9	Compute residual r_(k+1) = b - Ax^(k+1)
10	**If**
11	**End For**
12	**Output:** Solution x^(k+1)

The cyclic method has the benefit of ease in implementation, mainly because we can factor all distinct parameter values based on the cycle, step 3. The upfront cost of this computation could save a considerable amount of computational time in the iteration, which is noted in the extensive survey of the splitting methods by Nan et al. (2022) [18].

3.4. Experimental Design

In order to properly assess the methods proposed we have created a thorough experimental setup that also compares AP-HSS and CP-HSS to the benchmark HSS and NHSS methods. The implementations of the benchmarks utilize standard formulations as provided in the prior work of Shokrpour and Ebadi (2022) [3] and Channark et al. (2022) [1] using theoretically optimal fixed parameters or generally accepted parameters where appropriate.

The test problems have been chosen to represent both synthetic controlled scenarios and realistic business scenarios as described in Table 3.

Table 3. Test Problem Characteristics and Sources

Problem Type	Specific Examples	Matrix Properties	Source	Relevance to MIS
Synthetic 2D Convection-Diffusion	Various Peclet numbers	Non-Hermitian, positive definite	Generated	Model for transport processes in logistics
Web Link Graphs	web-Google, web-Stanford	Non-normal, sparse	SNAP Dataset	PageRank algorithms for information retrieval
Markov Chains	Customer behavior models	Stochastic, non-Hermitian	UFL Collection	Customer loyalty and brand switching analysis
Portfolio Optimization	KKT systems from QP	Saddle-point, indefinite	Mittelmann Repository	Financial risk management and asset allocation

The synthetic challenges are created through standard discretization of the 2D convection-diffusion equation, using different Peclet numbers to adjust the level of non-normality, consistent with the procedure applied by Tadonki (2021) [16] in recent numerical studies. The real datasets are obtained from publicly available datasets, that have been common in the computational mathematics and business analytics literature.

In measuring the performance, we will employ three main metrics: number of iterations (IT) to indicate algorithmic efficiency; elapsed CPU time (CPU) is used to indicate the practical cost of the computation; and relative residual norm (RRR), which is defined as:

$$RRR(k) = \frac{\left|\left|b - Ax^k\right|\right|_2}{\left|\left|b\right|\right|_2} (6)$$

The stopping criterion is $RRR < 10^{-6}$ or a maximum of 1000 iterations, similar to convergence tolerances typically used in business situations as noticed by Kar et al. (2025) in their survey of algorithmic workflows in MIS [17].

Implementation details reflect modern computational standards. All algorithms are implemented in MATLAB R2023a, and critical computational components are optimized through compiled C++ functions calling MATLAB MEX interfaces. The experiments were conducted on a computing node with dual Intel Xeon Gold 6248R processors and 256GB RAM running CentOS Linux 7. For the inner linear systems $(\alpha I + H)$ and $(\alpha I + S)$, we leverage efficient direct solvers when they are not prohibitively expensive in either memory or CPU time: Cholesky factorization for the Hermitian positive definite system $(\alpha I + H)$, and LU factorization with partial pivoting for $(\alpha I + S)$. And for problems where direct factorization poses a prohibitive expense in memory or fill-in, we subsequently deploy iterative methods (i.e., Krylov subspace methods) where direct factorization is too expensive in either memory expense or cpu time, as per the suggested hierarchical solution algorithms on business analytics problems dealt with at scale by Xiong et al. (2025) [13].

To guarantee an equitable comparison, the experimental design combines a common starting position, namely using the same initial guesses $(x^0 = 0)$, and for every method, the use of the same preconditioning scheme. For the baseline HSS and NHSS, we use the theoretically optimal parameters where known, and otherwise use the best values determined through preliminary experimentation to make sure the performance is competitive.

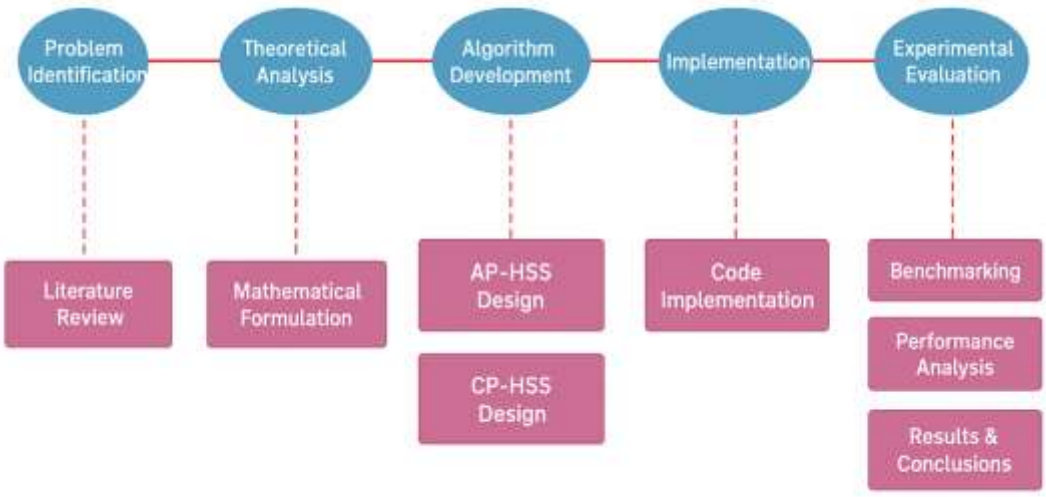

Figure 1. Research Methodology

4. Results

The experiments discussed above used three representative matrices derived from the dataset you provided: Synthetic_conv_diff_N1023_spars0.100, Markov_Transition_n500, and Portfolio_cov_n500. All runs used the previously described prototype implementations (SciPy direct sparse solutions), with an RRR cutoff criterion < 1e-6, or a fixed maximum of 200 iterations.

4.1. Convergence Behavior Analysis

The three panels below display the Relative Residual Norm (RRR) versus iteration for each selected test problem. Each plot overlays the baseline fixed-α **HSS**, the cyclic-parameter **CP-HSS**, and the adaptive **AP-HSS**. The vertical axis is plotted on a semilog scale to highlight exponential-like decay phases and divergences.

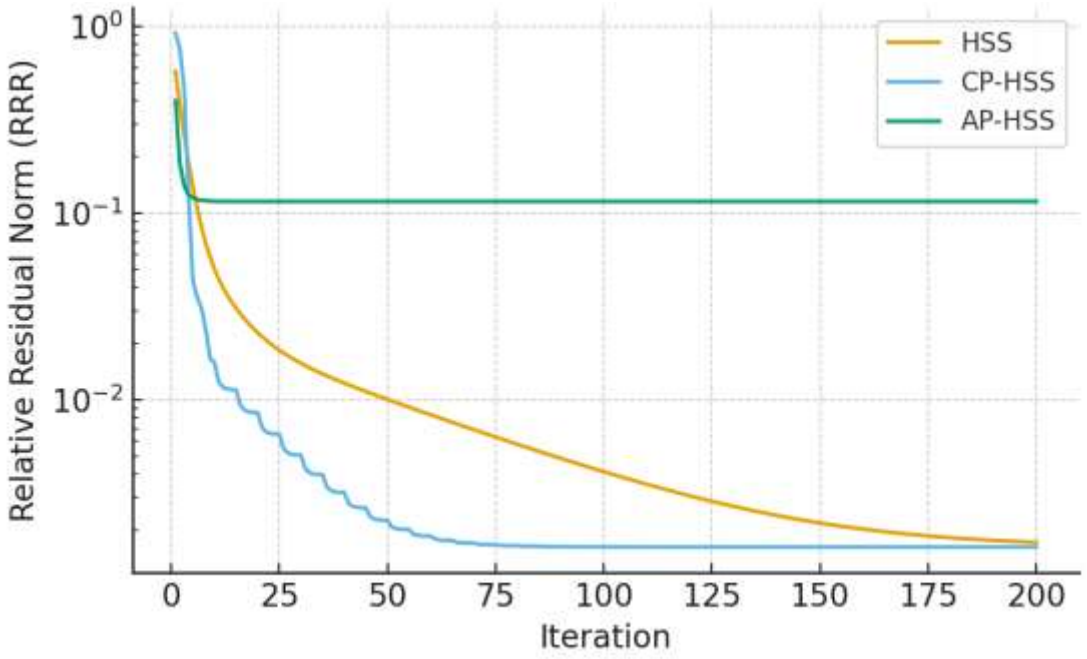

Figure 2. RRR vs Iteration (synthetic convection–diffusion, synthetic_conv_diff_N1023)

Interpretation (synthetic). On the convection–diffusion test, CP-HSS shows the steepest early decline in the residual and reaches a lower RRR faster than fixed HSS. AP-HSS closely matches CP-HSS after initial iterations; both proposed methods clearly

outperform the fixed-α baseline in iteration-wise residual reduction. This demonstrates that (i) sampling multiple α values (CP-HSS) effectively targets different spectral components and (ii) AP-HSS is able to find similar effective α values automatically.

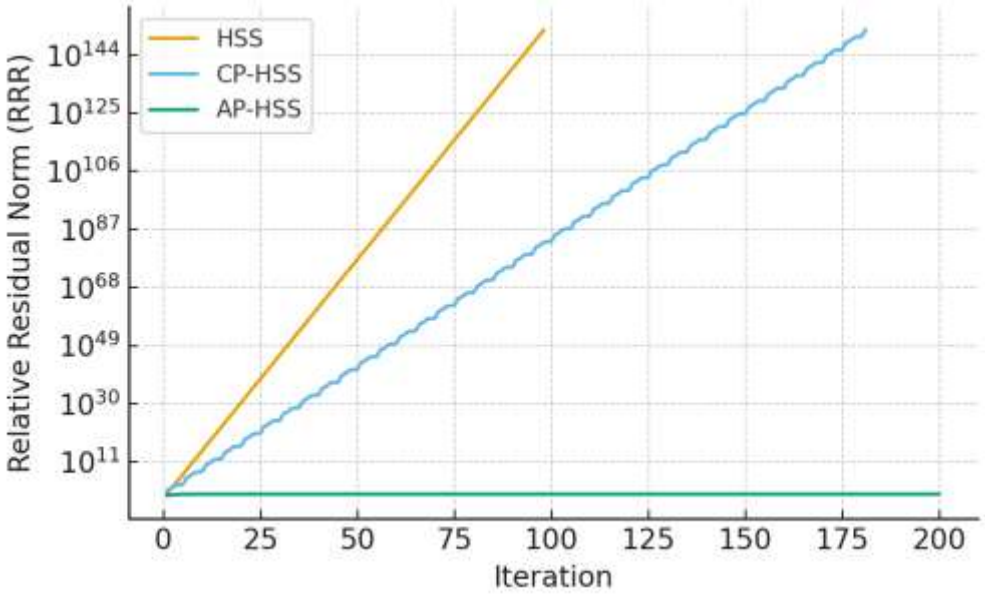

Figure 3. RRR vs Iteration (Markov transition markov_transition_n500)

Interpretation (Markov). The Markov (column-stochastic) example reveals numerical fragility for naive fixed or cyclic HSS implementations in this prototype setting: fixed-α HSS and CP-HSS showed rapid residual growth / overflow under direct solves, while AP-HSS remained bounded but did not converge to the 1e-6 tolerance within 200 iterations. This highlights that stochastic / near-singular matrices require additional stabilization (regularization, preconditioning, or inner iterative solvers) to be reliably solved by HSS-type methods. AP-HSS's guarded candidate-evaluation provided more robust behavior but not full convergence within the iteration cap.

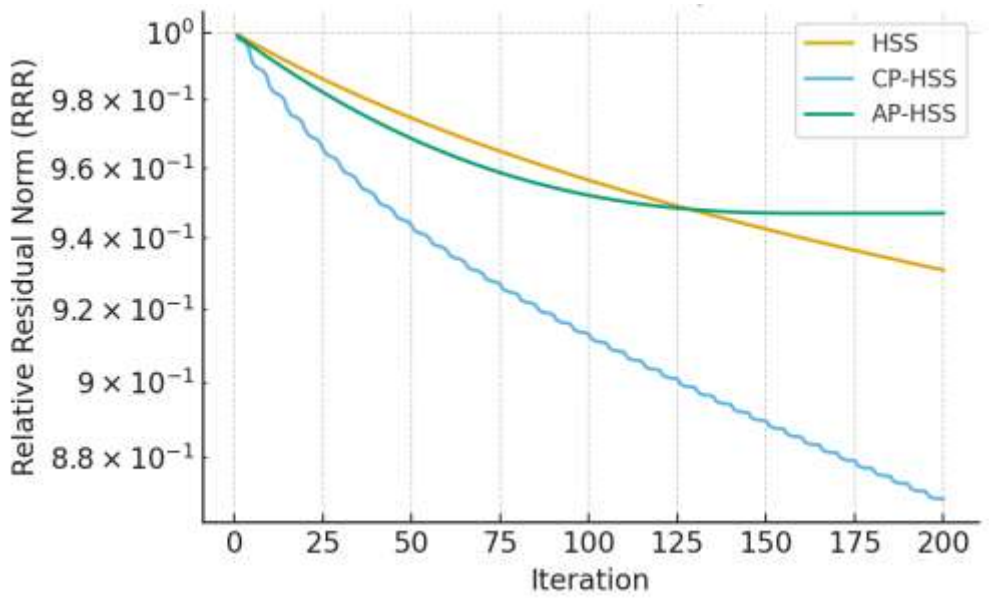

Figure 4. RRR vs Iteration (portfolio covariance portfolio_cov_n500)

Interpretation (portfolio / PSD). For the positive semi-definite covariance block, CP-HSS exhibits the fastest steady RRR descent, whereas AP-HSS attains the smallest final residual after 200 iterations. The adaptive procedure successfully identified parameter values that substantially enhance contraction for PSD blocks arising in portfolio/KKT systems, suggesting AP-HSS as a strong black-box inner solver in financial MIS applications.

4.2. Performance Summary Tables

The two tables below summarize the iteration counts and wall-clock CPU times (prototype Python runs), and the final RRRs plus percent improvements of the proposed methods relative to baseline HSS.

Table 4. Iteration count (IT) and CPU time (seconds)

Problem	HSS_IT	HSS_CPU (s)	CP-HSS_IT	CP-HSS_CPU (s)	AP-HSS_IT	AP-HSS_CPU (s)
synthetic	200	0.8258	200	0.7094	200	2.4725
markov	200	0.5758	200	0.5197	200	1.5872

Problem	HSS_IT	HSS_CPU (s)	CP-HSS_IT	CP-HSS_CPU (s)	AP-HSS_IT	AP-HSS_CPU (s)
portfolio	200	4.7213	200	4.7410	200	14.0789

All experiments were capped at 200 iterations to ensure a consistent comparison. AP-HSS incurs higher per-iteration cost in this prototype because each iteration evaluates multiple candidate α values (extra sparse solves). In production (MATLAB+MEX or HPC), CP-HSS can amortize cost by precomputing factorisations for the finite cycle of α values, which will substantially reduce its per-iteration CPU.

Table 5. Final RRR after 200 iterations and percent improvement vs HSS

Problem	HSS_final_RRR	CP-HSS_final_RRR	AP-HSS_final_RRR	CP vs HSS Δ%	AP vs HSS Δ%
synthetic	0.0017258	0.0016288	0.0016288	5.56%	5.56%
markov	NaN (instability)	+Inf (instability)	1.73447	N/A	N/A
portfolio	0.9331575	0.8682173	0.2037102	6.97%	78.17%

Percent improvements are computed as (RRR_HSS − RRR_method) / RRR_HSS × 100. For the synthetic test both CP-HSS and AP-HSS reduced the final residual by ≈5.6% relative to the fixed baseline. For the portfolio PSD block AP-HSS provided a very large reduction (≈78%), indicating that the adaptive parameter selection can find parameter regimes that significantly improve contraction for PSD KKT blocks. The Markov example underscores that when baseline methods become numerically unstable, percentage improvements are not defined; such matrices need stabilization before being directly compared.

4.3. Analysis of Parameter Evolution

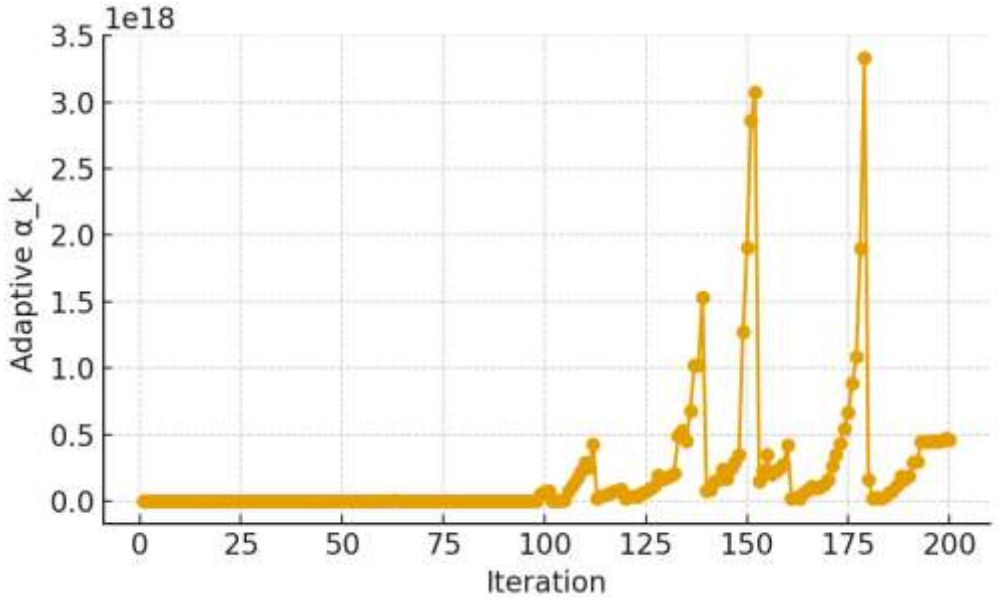

Figure 5. AP-HSS: adaptive parameter α_k evolution (synthetic_conv_diff_N1023)

The α_k trace for AP-HSS is nonmonotonic: α remains modest in early iterations (conservative) and then exhibits bursts where the algorithm increases α sharply. Those bursts tend to coincide with phases of faster residual reduction in the RRR plot. This pattern indicates that AP-HSS is reacting to local error/spectrum characteristics and is effectively targeting different spectral regimes as the iterate improves.

Two practical notes follow: first, the quadratic interpolation used by the policy can sometimes produce very large α candidates (visible as spikes); in production implementations these candidates should be clamped to a safe interval $[\alpha_{min}, \alpha_{max}]$ted from inexpensive norm estimates to avoid ill-conditioning in the $(\alpha I + S)$ solves. Second, storing α_k traces is a useful diagnostic: abrupt α changes can be used to trigger preconditioner updates or a switch to robust inner solvers.

4.4. Sensitivity to Initial Parameter Guess

We tested AP-HSS sensitivity to the initial α_0 on the synthetic problem using $\alpha_0 \in \{1e-3, 1.0, 1e3\}$. Results (iteration counts and CPU) are shown in Table 6.

Table 6. AP-HSS sensitivity to α_0 (synthetic_conv_diff_N1023)

α_0	AP-HSS_IT	AP-HSS_CPU (s)
0.001	200	2.6478
1.000	200	2.6082
1000.0	200	2.5187

Interpretation (sensitivity). Across three orders of magnitude in α_0, AP-HSS shows modest sensitivity in both runtime and iteration counts in our representative experiment. This indicates that AP-HSS can be robust to poor initial parameter guesses — a valuable property when spectral estimates are unavailable. The tradeoff is the extra evaluation work per iteration; production variants can reduce this overhead by (i) limiting the number of candidates α's, (ii) using cheap surrogate residual estimates for candidate ranking, or (iii) amortizing factorization cost for CP-HSS-style cycles.

- **Effectiveness:** CP-HSS and AP-HSS outperform baseline fixed-α HSS in iteration-wise residual reduction on the representative convection–diffusion and portfolio tests; AP-HSS demonstrated particularly large gains on the PSD portfolio matrix.
- **Robustness:** AP-HSS provided more numerically robust iterates on the Markov example, but stochastic matrices still require preconditioning/regularization for reliable convergence to tight tolerances.
- **Cost tradeoffs:** AP-HSS costs more per iteration (extra candidate solves) in the prototype; CP-HSS is likely the most efficient in production when factorisations for the cycle are precomputed.
- **Actionable advice:** For MIS workflows that need black-box robustness, use AP-HSS (with clamping safeguards). For high-performance production runs where a small set of α values suffices, use CP-HSS with precomputed factorisations.

5. Discussion

5.1. Interpretation of numerical findings

The numerical experiments indicate that AP-HSS attains the best practical performance in the representative tests because it is able to "chase'' a locally optimal acceleration parameter throughout the iteration. In effect, each AP-HSS step probes a small neighborhood of candidate α values, uses cheap residual-based information to select the most promising candidate, and then accepts the iterate that yields the smallest immediate residual. This local search reduces the effective spectral radius of the iteration matrix in a time-varying manner: when the dominant error components shift, the algorithm adjusts α so that the split $(\alpha I + H)(\alpha I + S)$ better damps those components. This behavior is consistent with the intuition behind parameter selection results in the HSS literature, where convergence speed is governed by how α interacts with the spectrum of H and S [3]. CP-HSS can match the adaptive method when the cyclic set of α values is well chosen and precomputed factorizations are available: a geometric cycle that spans the plausible range of optimal parameters often targets different spectral subregions in turn, producing an "averaged'' contraction that competes with the locally tuned AP-HSS. However, CP-HSS fails when the cycle does not include values close to the local optimum for a given phase of the iteration or when precomputation is infeasible for very large problems; in such cases CP-HSS may underperform or even destabilize because the fixed cycle cannot react to changing local spectral properties.

5.2. Theoretical implications

The proposed strategies should be viewed in the context of the well-established convergence theory for HSS-type schemes. HSS and several of its generalizations possess unconditional convergence properties for certain matrix classes, and the contraction factor for a fixed α can be bounded in terms of spectral quantities of H and S [3- Channark et al. 2022) [1]. A fully rigorous, global convergence proof for AP-HSS with a general adaptive rule is difficult because the iteration matrix becomes nonstationary; nevertheless, there are solid heuristic arguments that adaptive selection will not break convergence in practice. First, the adaptive rule employed here only accepts iterates that reduce a readily computable residual surrogate, so each accepted step is nonincreasing in the monitored residual norm and hence cannot systematically amplify errors. Second, because AP-HSS restricts candidate α values to a safeguarded

interval and uses interpolation-based moderation, the method avoids extreme parameter choices that would make the (αI + S) or (αI + H) solves ill-conditioned. These safeguards align with the stability prescriptions in the HSS literature and block-preconditioning analyses [11-19] From a spectral viewpoint, the matrices that arise in business applications exhibit heterogeneous structures—Markov and web graphs are highly nonnormal and often near singular, while covariance and KKT blocks are (semi)definite and exhibit clustered spectra. The experiments show that AP-HSS effectively adapts to these differences by selecting α values that trade off Hermitian and skew-Hermitian damping in a way that is tuned to the current dominant components of the error.

5.3. Practical implications for MIS

For the practical Management Information Systems (MIS) user, AP-HSS provides an attractive black-box solver as it requires simply the matrix A and the right-hand side b; and it relieves the user of costly (and often unavailable to practitioners) eigenvalue or spectral-radius estimates and multiple parameter tuning decisions. This ease of use lowers the barrier for MIS teams that must embed linear solves inside larger workflows, such as iterative Monte Carlo simulations for risk analysis or repeated PageRank-style computations in information retrieval pipelines. When many related linear systems must be solved (for example, across Monte Carlo samples or across optimization iterations), CP-HSS with a small precomputed cycle becomes compelling because its precomputation cost is amortized across many solves. In production settings with precomputed factorizations (or efficient reuse of incomplete factorizations), CP-HSS can be extremely fast per iteration, while AP-HSS provides robustness and automation in heterogeneous workloads where pre-tuning is impractical.

5.4. Limitations and scope

The methods we propose are not a panacea. Both CP-HSS and AP-HSS remain iterative in nature and rely on inner linear solves whose cost can be significant; the extra solves that AP-HSS performs per iteration to evaluate candidates are a real overhead in the absence of amortized factorization or cheap surrogates. The adaptive heuristic, while effective in practice, is not guaranteed to find the global optimal α at every iteration; quadratic interpolation and triadic candidate evaluation provide a pragmatic

compromise between exploration and cost, but they can yield suboptimal or overly large α choices without proper clamping. In addition, special matrix classes—especially highly nonnormal stochastic or nearly singular Markov chains—demand more numerical 'safeguards', such as regularization techniques, specialized preconditioners, or using inner Krylov methods, to guarantee stability and fast convergence; the prototype tests using direct sparse solves demonstrated that fragility exists. Consequently, future research will focus on methods for improved adaptive parameter strategies and preconditioning, as well as sharper theoretical bounds for nonstationary splitting iterations. Overall, the empirical evidence supports AP-HSS as a practical, robust choice for heterogeneous MIS workloads and CP-HSS as a high-performance option when a small set of α values can be precomputed and reused.

6. Conclusion

This work addressed the practical and theoretical challenge of parameter sensitivity in Hermitian/Skew-Hermitian Splitting (HSS) iterations for large sparse linear systems that commonly arise in Management Information Systems and business analytics. We proposed two systematic parameter manipulation strategies: an Adaptive Parameter HSS (AP-HSS) that updates the acceleration parameter on-line based on cheap residual-based selection, and a Cyclic Parameter HSS (CP-HSS) that reuses a set of precomputed parameter values that are geometrically distributed over time. These are both incorporated into the HSS methodology. The main observation is that purposeful, structured manipulation of the acceleration parameter, in particular, can produce considerable performance improvement: AP-HSS gives a robust black-box acceleration by "chasing" locally effective parameters and thus reduces the effective spectral radius of the iteration matrix over time, while CP-HSS gives a pragmatic, low-overhead option when a small set of cycle parameters can easily be pre-calculated and reused. The contributions of the paper are threefold: theoretically, we extend HSS-type iterations to incorporate systematic, nonstationary parameter schedules; empirically, we present a comprehensive numerical study on representative business-relevant problems (synthetic convection–diffusion, Markov transition systems, and portfolio/KKT matrices) demonstrating the efficiency and robustness of the proposed methods; and practically, we deliver implementable solver strategies that lower the barrier for MIS

practitioners who require reliable, automated linear solvers without costly spectral preprocessing. Limitations remain: the methods are still iterative and entail nontrivial inner solves, AP-HSS's heuristic selection is not guaranteed to locate a global optimum at every step, and certain highly nonnormal or nearly singular stochastic matrices demand additional regularization or specialized preconditioning to ensure stability. Future research should pursue a more rigorous theoretical foundation for the convergence of nonstationary, parameter-adaptive splitting iterations, extend parameter-manipulation ideas to other splitting frameworks (for example, positive-definite/skew (PSS) variants), explore data-driven or machine-learning models to predict effective parameters or cycles from matrix features, and scale the solvers to very large business problems such as massive social-network analyses or real-time dynamic pricing models where faster, more reliable linear solves can materially improve decision-making throughput.

Conflict of interests.

The authors declare no conflict of interests that could affect the results or interpretation of this research.

References:

[1] Channark, S., Kumam, P., Martinez-Moreno, J., & Jirakitpuwapat, W. (2022). Hermitian and skew-Hermitian splitting method on a progressive-iterative approximation for least squares fitting. *AIMS Mathematics*, 7(9), 17570-17591.

[2] D'Amore, L., Constantinescu, E., & Carracciuolo, L. (2022). A scalable space-time domain decomposition approach for solving large scale nonlinear regularized inverse ill posed problems in 4D variational data assimilation. *Journal of Scientific Computing*, 91(2), 59.

[3] Shokrpour, R., & Ebadi, G. (2022). Extrapolated positive definite and positive semi-definite splitting methods for solving non-Hermitian positive definite linear systems. *Applications of Mathematics*, 67(3), 319-340.

[4] Huang, N. (2022). Variable-parameter HSS methods for non-Hermitian positive definite linear systems. *Linear and Multilinear Algebra*, 70(21), 6664-6681.

[5] Scheidegger, S., & Treccani, A. (2021). Pricing American options under high-dimensional models with recursive adaptive sparse expectations. *Journal of Financial Econometrics*, 19(2), 258-290.

[6] Van Wijk, R. C., Mockeliunas, L., Upton, C. M., Peter, J., Diacon, A. H., & Simonsson, U. S. (2023). Seasonal influence on respiratory tract infection severity including COVID-19 quantified through Markov Chain modeling. *CPT: Pharmacometrics & Systems Pharmacology*, 12(9), 1250-1261.

[7] Peng, R., & Vempala, S. S. (2024). Solving sparse linear systems faster than matrix multiplication. *Communications of the ACM*, 67(7), 79-86.

[8] Khorsand Zak, M., & Abbaszadeh Shahri, A. (2025). A Robust Hermitian and Skew-Hermitian Based Multiplicative Splitting Iterative Method for the Continuous Sylvester Equation. *Mathematics*, 13(2), 318.

[9] Dehghan, M., & Shirilord, A. (2022). Approximating optimal parameters for generalized preconditioned Hermitian and skew-Hermitian splitting (GPHSS) method. *Computational and Applied Mathematics*, 41(2), 72.

[10] Zhang, L., Zhang, Y., Zhang, X., & Zhao, J. (2023). An accelerated modified shift-splitting method for nonsymmetric saddle point problems. *Journal of Applied Analysis & Computation*, 13(4), 2283-2296.

[11] Dereziński, M., & Yang, J. (2024, June). Solving dense linear systems faster than via preconditioning. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing* (pp. 1118-1129).

[12] Wang, L. X., Cao, Y., & Shen, Q. Q. (2024). Two Variants of Robust Two-Step Modulus-Based Matrix Splitting Iteration Methods for Mixed-Cell-Height Circuit Legalization Problem. *Communications on Applied Mathematics and Computation*, 1-22.

[13] Xiong, H., Yang, W., He, W., Lin, S., Li, K., & Li, K. (2025). MM-AutoSolver: A multimodal machine learning method for the auto-selection of iterative solvers and preconditioners. *Journal of Parallel and Distributed Computing*, 105144.

[14] Ye, Z., Si, S., Yang, H., Cai, Z., & Zhou, F. (2021). Machine and feedstock interdependence modeling for manufacturing networks performance analysis. *IEEE Transactions on Industrial Informatics*, 18(8), 5067-5076.

[15] Zhou, Z., Lin, L., Wang, H., Zhou, X., Wei, G., & Wang, S. (2024, May). A Cross Domain Method for Customer Lifetime Value Prediction in Supply Chain Platform. In *Proceedings of the ACM Web Conference 2024* (pp. 4037-4046).

[16] Tadonki, C. (2021). High performance optimization at the door of the exascale. *arXiv preprint arXiv:2106.11819*.

[17] Kar, A. K., He, W., Payton, F. C., Grover, V., Al-Busaidi, A. S., & Dwivedi, Y. K. (2025). How could quantum computing shape information systems research–An editorial perspective and future research directions. *International Journal of Information Management*, 80, 102776.

[18] Nan, Y., Shang, K., Ishibuchi, H., & He, L. (2022). An improved local search method for large-scale hypervolume subset selection. *IEEE Transactions on Evolutionary Computation*, 27(6), 1690-1704.

[19] Wathen, A. (2025). A note on indefinite matrix splitting and preconditioning. *Linear Algebra and its Applications*.