Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

PRECONDITIONING A RESTARTED GMRES SOLVER USING THE RANDOMIZED SVD

José A. Moríñigo[1], Andrés Bustos[1], and Rafael Mayo-García[1]

[1]Dept. de Tecnología, Centro de Investigaciones Energéticas, Tecnológicas y Medioambientales,
Avda. Complutense 40, CIEMAT, Madrid, SPAIN

ABSTRACT

The performance of a CPU-only implementation of the restarted GMRES algorithm with direct randomized-SVD -based preconditioning has been analyzed. The method has been tested on a set of sparse and dense matrices exhibiting varying spectral properties and compared to the ILU(0) -based preconditioning. This comparison aims to assess the advantages and drawbacks of both approaches. The trade-off between iteration-to-solution and time-to-solution metrics is discussed, demonstrating that the proposed method achieves an improved convergence rate in terms of iterations. Additionally, the method's competitiveness with respect to both metrics is discussed within the context of several relevant scenarios, particularly those where GMRES-based simulation techniques are applicable.

1 INTRODUCTION

The need to improve both direct and iterative solvers has become increasingly crucial at the dawn of the exascale HPC era, as extremely large systems of linear equations require accurate solutions (Agullo et al. 2021). This challenge can be viewed both in isolation and as an integral component of more complex solvers. In either case, it necessitates addressing multiple algorithmic fronts simultaneously. Among these, the present study focuses on accelerating convergence by enhancing the preconditioning step (Benzi 2002), thereby enabling iterative solvers to reach solutions within a reasonable time frame. Specifically, Krylov subspace-based solvers, when paired with appropriate preconditioners, are generally considered an optimal choice for achieving both robustness and efficiency (Moríñigo et al. 2022; Eldén and Simoncini 2012; van der Vorst 2003; Vuik 2018). The Generalized Minimum RESidual (GMRES) method and its variants (Saad and van der Vorst 2000; Saad 2020; Saad 1993; Saad and Schultz 1986; Li and Saad 2013; Vuik 1995) serve as the foundation for this study.

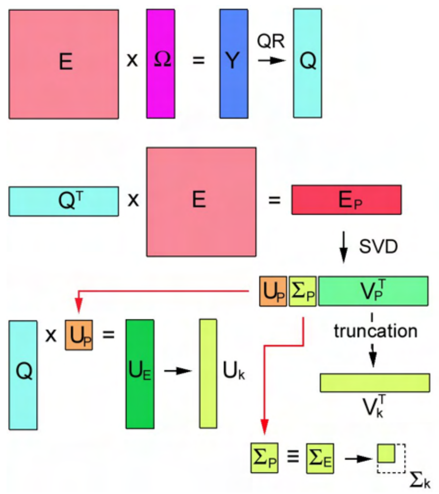
This work evaluates the performance of a novel preconditioner integrated into the standard restarted GMRES algorithm, which is highly used in many science-based simulations. A CPU-only serial implementation has been developed, and its results are analyzed. The primary contribution of this study is to provide a clearer understanding of the advantages and limitations associated with this class of preconditioners. For this purpose, the C programming language has been employed. The analysis is conducted using a set of sparse and dense matrices of varying size and condition number.

This article is organized as follows. The next section provides a brief overview of the randomization approach combined with the SVD formulation, which leads to the randomized SVD (rSVD) algorithm based on performing an economical truncated-SVD. This rSVD corresponds to the so-called 'direct rSVD' (Higham and Mary 2019; Martinsson 2019). Section 3 presents the implementation of this preconditioner in a restarted GMRES solver, utilizing an error matrix as the basis for the proposed improvement. Subsequently, the numerical experiments conducted with a range of sparse and dense matrices are summarized. Section 5 presents the results obtained from the implementation, evaluating its performance in terms of iteration-to-solution and time-to-solution metrics. Section 6 contextualizes this work within the framework of related research. Finally, conclusions are drawn.

2 RANDOMIZED SVD

The computational cost of the SVD in large scale problems suggests the benefits of exploiting randomization techniques (Halko et al. 2009; Martinsson 2019) to mitigate it. The key point is to efficiently find a low-rank approximation of the involved large, dense matrices, thereby accelerating the algorithm. In general, applying the standard deterministic SVD algorithm to a large matrix incurs high CPU and memory costs. Consequently, strategies for reducing these costs are of particular importance. The randomized SVD (rSVD) approach exemplifies such a strategy, wherein a small random subspace is constructed, and the original large matrix is projected onto this subspace. This projection effectively compresses the primary information of the large matrix into it, yielding a low-rank approximation of the original matrix and so reducing the computational cost of the SVD. These seminal ideas were first introduced by Halko et al. (2009) over a decade ago.

The rSVD algorithm incorporated into the GMRES method is described in what follows. Specifically, rSVD operates on the $n \times n$ error matrix E (see next section), such that a $k \times k$ matrix E_k is built as a truncated SVD of E. The steps are shown in Figure 1 (graphically and with pseudocode, left and right sides respectively). Basically, instead of applying a deterministic SVD to the target matrix E, its columns are sampled using a random Gaussian matrix Ω to build the projected matrix Y. This process ensures that the columns of Y span a lower-rank subspace that, with high probability, captures the k dominant singular triplets (singular values and vectors) of the target matrix E. The projection space defined by matrix Ω spans k dimensions, which are augmented by a small oversampling parameter p to ensure that the columns remain linearly independent with high probability. Thus, $\Omega \in \mathbb{R}^{n\times(k+p)}$, being $p < 10$ in practice, with $k+p \leq n$ in a strict sense (see Halko et al. (2009)). Typically, Y and Q are tall-skinny matrices whose number of columns fulfills $k+p \ll n$ in most cases of interest. The orthonormal basis Q captures the range of E, enabling the computation of a k-rank version of $Q^T E$ via a deterministic truncated SVD.



```
Pseudocode  Randomized SVD - via direct SVD of $Q^T E$
Input: $E \in \mathbb{R}^{nxn}, k \in \mathbb{N}$ and $k+p \leq n$
Output: $k$-rank truncated randomized SVD
        factors $U_k \in \mathbb{R}^{n\times k}, \Sigma_k \in \mathbb{R}^{k\times k}$ and $V_k \in \mathbb{R}^{n\times k}$
  1. Build $n \times (k+p)$ random matrix $\Omega$
  2. $Y = E\Omega$
  3. Compute an orthogonal basis: $Q = qr(Y)$
  4. Projection on random subspace: $E_P = Q^T E$
  5. $[U_P, \Sigma_P, V_P] = svd(E_P)$
  6. $U_P \leftarrow QU_P$
  7. $U_k = U_P(:,1:k), \Sigma_k = \Sigma_P(1:k,1:k), V_k = V_P(:,1:k)$
  8. The truncated SVD of $E$ is: $E_k = U_k \Sigma_k {V_k}^T$
```

Figure 1: Randomized SVD algorithm: QR-orthogonalization, deterministic SVD on a smaller matrix, and further truncation to the leading singular triplets are indicated in the sketch (left side). Pseudocode is provided on the right side. Notice that the tall-skinny matrix Ω is instrumental in reducing the entire computing cost.

As previously highlighted, computing the full spectrum of singular values and vectors can significantly increase the overall computational cost. Therefore, the method described here exploits the truncation of the spectrum to the k top singular triplets using the randomized SVD approach. In the present study, the largest portion of the spectrum considered spans up to 30%. This limit is set to control the high computational cost of the SVD on large matrices, which would otherwise lead to overhead in constructing the GMRES preconditioner. In addition to k, the oversampling parameter p also drives the cost of constructing the low-rank approximation of E. From a computational efficiency perspective, the ideal scenario is to select $k+p \ll n$, then Ω, Y and Q are tall-skinny matrices. It should be noticed that the overall performance of the rSVD algorithm is driven by the efficiency of matrix-matrix multiplications, QR factorization and the deterministic SVD performed in a much smaller matrix to drop the computing cost. The impact of the parameters k, p on the GMRES performance will be analyzed in the results section.

3 ALGORITHM IMPLEMENTATION

In many large-scale problems, especially those arising from discretized partial differential equations, the matrix condition number is often large, which leads to slow convergence. Preconditioning aims to transform the system into one with a more favorable condition number, thereby reducing the iterations required to converge, particularly for Krylov subspace methods like the Conjugate Gradient or Generalized Minimal RESidual (GMRES). Furthermore, effective preconditioning can help mitigate the effects of round-off errors and improve the robustness of the solver. It is noticed that preconditioning not only improves performance but also contributes to the overall reliability of numerical simulations and equation systems solving.

3.1 Preconditioner Building

A widely used iterative method for solving the linear system $Ax = b$, where the matrix $A \in \mathbb{R}^{n \times n}$ is non-singular, non-symmetric, and typically sparse, is the restarted GMRES method (Saad and Schultz 1986; Saad 1993; Saad 2020; Saad and van der Vorst 2000; Vuik 1995). This method often incorporates a preconditioner to accelerate convergence to the solution. Typically, the preconditioner exploits an approximate LU-factorization of A, aiming to reduce the fill-in of the factors. This approach has been considered a reasonable trade-off, though it remains suboptimal, as convergence may still be slow.

Among the various approximate LU-factorizations, the zero-order approximation ILU(0) has been extensively implemented in many solvers. Applied to a sparse matrix A, it produces the approximate factors $\hat{L}, \hat{U}$, which preserve the sparsity pattern of A, making it competitive in terms of computational cost. In general, an approximate factorization of matrix A can be expressed as $A = \hat{L}\hat{U} + \Delta A$, leading to the introduction of the error matrix $E = M^{-1}\Delta A$, where the preconditioner $M = \hat{L}\hat{U}$ is derived from the ILU(0) factorization. It is important to note that this error matrix is linked to the deviation between the exact and approximate factorizations of A.

A key property linking matrix A and error matrix E holds: if A is ill-conditioned (i.e., $\kappa(A) = \|A\|\|A^{-1}\| \gg 1$), then matrix E retains a low-rank structure numerically, meaning it has only a few large singular values (Higham and Mary 2019). This structure can be exploited to modify the LU-based preconditioning, improving the low convergence rate that ILU(0) alone often yields in many cases. Notably, high $\kappa(A)$ is not a strict requirement for this approach, suggesting that the formulation has a wide scope. This reformulation of the LU-based preconditioning uses a sketch of the error matrix $E = \hat{U}^{-1}\hat{L}^{-1}A - I$, denoted as E_k. The matrix E_k is obtained by applying the randomized SVD algorithm, as outlined in the previous subsection (see pseudocode in Figure 1).

In summary, the proposed approach substitutes the ILU(0) preconditioner $M^{-1} = \hat{U}^{-1}\hat{L}^{-1}$ with the novel formulation $M^{-1} = (I + E_k)^{-1}\hat{U}^{-1}\hat{L}^{-1}$, where E_k represents the k-rank approximation to the error matrix E. The degree of fidelity to the original error matrix is controlled by parameter k. The correction factor $(I + E_k)^{-1}$, which premultiplies the standard preconditioner $\hat{U}^{-1}\hat{L}^{-1}$, leads to accelerated convergence. In the ideal case, when $E_k = E$, the preconditioner becomes $M^{-1} = A^{-1}$. It is intuitive to see that if k is of order n, the preconditioner will be more robust but will not be cost-competitive compared to ILU(0). Furthermore, when $k \ll n$, the expression $I + E_k$ can be computed efficiently using the Sherman-Morrison-Woodbury formula (Henderson and Searle 1981). Since E is often low-rank numerically, a small k is likely to provide a good approximation to E, making the preconditioner close to the ideal case. These ideas are tested in

Algorithm 1 Restarted GMRES(m) with Preconditioning

Input: $A, M \in \mathbb{R}^{nxn}, b \in \mathbb{R}^n$, initial guess $x_0 \in \mathbb{R}^n; m, k, p \in \mathbb{N}$ with $k + p \leq n$

Set preconditioner: $M := \hat{L}\hat{U}(I + E_k)$, being $\hat{L}, \hat{U}$ the ILU(0) triangular factors of A, and $E_k \approx E$ is a truncated-SVD approximation with $k \ll n$

Output: Solution $x \in \mathbb{R}^n$ for the system $Ax = b$

```
1. Repeat: restart-loop ($x_0$)
2. Arnoldi process
   $r_0 := b - Ax_0, \beta := \|r_0\|_2, v_1 := r_0/\beta$
   for $j = 1, 2, ...m$ do
      $z_j = M^{-1} v_j$
      $w = Az_j$
      for $i = 1, 2, ...j$ do
         $h_{i,j} := w \cdot v_i$
         $w := w - h_{i,j} v_i$
      end for
      $h_{j+1,j} := \|w\|_2$
      $v_{j+1} := w / h_{j+1,j}$
      $Z_m := [z_1, z_2, ...z_m]$
      $H_m := h_{i,j}$, being $1 \leq i \leq j+1; 1 \leq j \leq m$
   end for
3. Form the approximate solution
   $y_m := argmin_y \|H_m y - \beta e_1\|_2$, with $e_1 = [1, 0, ...0]$
   $x_m := x_0 + Z_m y_m$
4. Restart: If converged return $x_m$, else set $x_0 := x_m$ and go to step 1
```

the current study with a preconditioned restarted GMRES method outlined in Algorithm 1, implemented for CPU-only computing.

3.2 Parameters Setting

In the GMRES setting, vector b in the system $Ax = b$ is initialized by setting the solution vector x to all ones, except for its last element, which is set to n. The system is then solved by computing $b = Ax$, ensuring a valid solution. The inner-loop constructs a Krylov subspace with $m = 20$ direction vectors, and the outer-loop is restarted twenty times, allowing a maximum of 400 iterations. Once this value is reached, a non-convergence flag is raised by the solver, and the iteration stops. GMRES convergence is evaluated using the equations' residual as the iteration progresses. A stopping criterion based on both absolute and relative residual Euclidean norms is checked after each inner-loop iteration, and the process halts when these residuals drop below the prescribed tolerance, set to 10^{-8}. It is noted that this tolerance is applied uniformly across all matrices analyzed, regardless of their condition number. No attempt has been made in the present study to adjust the tolerance based on the matrix conditioning, which is a subject for future investigation.

4 NUMERICAL EXPERIMENTS

A set of 36 square matrices, both sparse and dense, with a broad range of condition numbers has been selected for the numerical experiments.

4.1 Sparse PDE-derived Matrices

Eighteen sparse, non-symmetric, square matrices (see Table 1 for the full list) have been selected from the widely used SuiteSparse Matrix Collection (Davis and Hu 2011), many of which are derived from real-world applications. These matrices have been previously analyzed in the context of fault-tolerant solvers (Moríñigo et al. 2022). The matrices represent problems mainly from fluid dynamics and a few from electromagnetism. They are classified into three groups based on their condition number $\kappa(A)$: 6 matrices with low $\kappa(A) \sim O(1) - O(10^2)$; 6 matrices with high $\kappa(A) \sim O(10^3) - O(10^4)$; and 6 matrices with very high $\kappa(A) \sim O(10^6) - O(10^{12})$. It is emphasized that the selected matrices avoid symmetry, as the Conjugate Gradient (CG) method is generally more efficient than GMRES for symmetric problems. Therefore, the matrices correspond to strictly non-symmetric problems. The SuiteSparse matrices are read in compressed format, storing only non-zero entries. A matrix reader, implemented in C, handles the data input for the GMRES implementation.

4.2 Dense Synthetic Matrices

Eighteen dense, non-symmetric, square matrices have been synthetically generated using the SVD factorization $A = U\Sigma V^T \in \mathbb{R}^{n\times n}$, where $U, V \in \mathbb{R}^{n\times n}$ are random orthogonal matrices, and the diagonal matrix $\Sigma = [\sigma_{ij}] \in \mathbb{R}^{n\times n}$, with $\sigma_{ij} = 0$ for $i \neq j$, has a prescribed singular value distribution based on a given $\kappa(A)$. Three condition numbers, $\kappa(A) = 10^2, 10^5$, and 10^8, are investigated. The dense matrices are built to exhibit either exponentially or uniformly decaying singular value spectra, with the decay profiles tailored to ensure that the resulting matrices attain the prescribed condition numbers. The matrix sizes considered are $n = 250, 1000, 4000$. The combination of $\kappa(A)$, singular value distribution, and matrix size results in the 18 dense matrices used in this study. The matrix nomenclature is as follows: $n\$SIZE\_\$LAW\_\$COND$, where $SIZE \in \{250, 1000, 4000\}$, $LAW \in \{\text{uni}, \text{exp}\}$, and $COND \in \{\text{cond1e2}, \text{cond1e5}, \text{cond1e8}\}$. For example, the matrix $n1000\_exp\_cond1e5$ corresponds to a 1000×1000 dense matrix with exponentially decaying singular values and $\kappa(A) = 10^5$.

4.3 Executions and HPC Facility

The execution of the randomized-SVD-based GMRES algorithm was performed on the Turgalium cluster at CETA-CIEMAT. Computations were carried out using a CPU partition comprising 20 nodes, each equipped with two Intel Gold 6254 CPUs (18 cores/CPU) running at 3.1 GHz, alongside 192 GB of DDR4 memory per node. The GMRES binary was compiled using the C compiler of the NVIDIA HPC SDK v21.9 suite. The execution setup, submission to the cluster's batch system, and subsequent post-processing of metrics were fully automated via Python scripts. For each matrix A, fifty repetitions of the GMRES execution have been carried out for every combination of oversampling p and $rank$ parameters. The substantial number of runs of the serial code required to elaborate the performance maps justifies the use of HPC resources. Performance maps have been plotted as functions of the parameters $[p - rank/n]$, within the ranges $p \in [1, 10]$ and $rank/n \in [0, 0.3]$, where $rank$ is better expressed as a fraction of the matrix size n. The maps include the mean and standard deviation of both time-to-solution and iteration-to-solution count, which are the performance metrics considered. Due to space constraints, only the most relevant maps are presented in the results section.

5 RESULTS

5.1 Minimum Iterations to Solution

An important question concerns the number of iterations required for convergence when using an rSVD-based preconditioner. Tables 2 and 3 present the iteration-to-solution metric for sparse and dense matrix sets, respectively. Truncated SVDs retaining 2% and 30% of the singular triplets, along with ILU(0)-based preconditioning, are compared to evaluate the convergence. As expected, retaining more singular triplets

Table 1: Square, non-symmetrical matrices from the SuiteSparse Matrix Collection, grouped by $\kappa(A)$.

Matrix	Size	$\kappa(A)$	Application
Low condition number: $\kappa(A) \sim O(10) - O(10^2)$			
dw256B	512 × 512	3.7	Electromagnetism
pde225	225 × 225	39	Model partial differential equation
cdde4	961 × 961	68	Convection–diffusion model
poisson2D	367 × 367	133	Poisson PDE in two-dimension
ex37	3,565 × 3,565	180	Computational fluid dynamics
bfwa62	62 × 62	553	Electromagnetism
High condition number: $\kappa(A) \sim O(10^3) - O(10^4)$			
bwm200	200 × 200	$2.4 \cdot 10^3$	Chemical processes
ex1	216 × 216	$3.3 \cdot 10^4$	Computational fluid dynamics
ex22	839 × 839	$3.3 \cdot 10^4$	Computational fluid dynamics
orsirr_2	886 × 886	$6.3 \cdot 10^4$	Computational fluid dynamics
tub100	100 × 100	$1.3 \cdot 10^4$	Tubular reactor model
olm100	100 × 100	$1.5 \cdot 10^4$	Computational fluid dynamics
Very high condition number: $\kappa(A) \sim O(10^6) - O(10^{12})$			
pores_1	30 × 30	$1.8 \cdot 10^6$	Computational fluid dynamics
steam1	240 × 240	$2.8 \cdot 10^7$	Oil-Steam modelling
DK01R	903 × 903	$5.9 \cdot 10^7$	Turbulent flow
saylr1	238 × 238	$7.9 \cdot 10^8$	Computational fluid dynamics
steam3	80 × 80	$5.0 \cdot 10^{10}$	Oil-Steam modelling
lung1	1,650 × 1,650	$4.9 \cdot 10^{12}$	Bio-fluid mechanics - multiphysics

(higher $rank/n$ in the truncation stage) reduces iteration count, though at the cost of increased computational time due to SVD building. Thus, a low $rank/n$ is desirable for cost-effective preconditioning. The data show that even minimal spectral content leads to improved convergence over ILU(0). It is worth noting that certain considerations out of scope of the present work but related to attain improved scalability of rSVD for very large matrices (Ferrero-Roza et al. 2024; de Castro-Sánchez et al. 2025), motivate the focus on randomized rather than deterministic SVDs. Performance comparison between rSVD and ILU(0) is shown in Fig. 2 for sparse matrices across $rank/n \in [0.02, 0.3]$. Most sparse matrices benefit from rSVD-based preconditioning. Comparable benefits are observed within the dense matrix subset reported in Table 3, which includes only those cases with the highest iteration counts to convergence (dense matrices requiring fewer than four iterations with ILU(0)-based preconditioning are excluded). In very small matrices such as $bfwa62$ or $pores\_1$, setting $rank/n = 0.02$ often results in $rank = 1$, i.e., only the leading singular value is used. Likewise, $olm100$ and $tub100$ ($n = 100$) use only their first two singular values. Despite the minimal information retained, iteration counts improve. However, matrices that already converge rapidly with ILU(0) (e.g., $steam3$ converging in 3 iterations) distort the rSVD comparison, as each extra iteration has a large relative impact. This also occurs with some dense matrices, which should be interpreted with care due to the magnified effects at very low iteration counts.

5.2 Time to Solution

This subsection evaluates execution time, addressing whether rSVD-based preconditioning is worthwhile. Two aspects are key: (i) the cost of the preconditioner relative to total runtime (noting the high cost of SVD), and (ii) scenarios where GMRES with rSVD-based preconditioning outperforms alternatives like ILU(0). For the first point, Figure 3 shows the ratio of preconditioner build time to total GMRES runtime for different $rank/n$ values (sparse and dense cases). The total runtime comprises the SVD cost (t_{SVD}) and

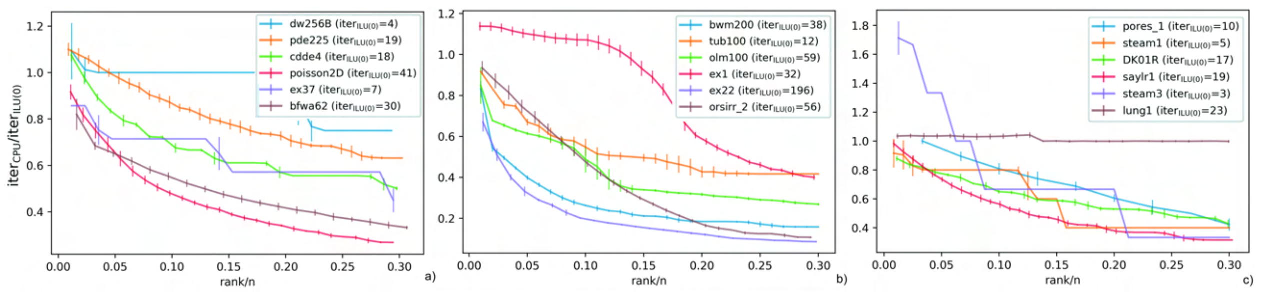


Figure 2: Iteration-to-solution of the GMRES preconditioned with rSVD, applied to the sparse matrix set: a) low $\kappa(A)$; b) high $\kappa(A)$, and c) very high $\kappa(A)$. Mean value and range of variation are plotted.

the Krylov iterations (t_{Krylov}), with t_{SVD} forming the dominant share of the preconditioner build time (t_{PC}). The red curve for $rank/n \in (0, 0.05)$ is of particular interest, reflecting low-cost preconditioners based on limited spectral data (which is a desirable setup for practical problems). Interestingly, the sparse matrices with the highest Krylov iteration counts, such as $ex22$ (131), $orsirr\_2$ (52), $poisson2D$ (38), and $ex1$ (36), show the lowest relative preconditioning cost. For dense matrices, the lowest relative preconditioning cost is observed for those with $\kappa(A) = 10^8$, particularly when the singular values decay exponentially. These findings suggest that rSVD-based preconditioning is most effective in scenarios characterized by slow convergence or very high condition numbers.

Table 4 presents the Krylov iteration time ratio (rSVD vs ILU(0)) at $rank/n = 0.3$ and 0.05, highlighting performance bounds. As anticipated, more spectral information speeds convergence due to reduced Krylov iterations, though it increases preconditioner build time. This trade-off is worthwhile under three conditions: when the rSVD assembly satisfies $rank/n \ll 1$, making SVD construction relatively cheap; when solving hard-to-converge problems; and in multi–right-hand-side cases ($AX = B$ where X and B are matrices with m columns). In the latter, the preconditioner is built once and used to solve the m systems, improving competitiveness. Figure 4 presents the computed metrics along with their corresponding standard deviations (std) for four representative sparse matrices. The std values exhibit only minor variations across the respective maps, indicating that fifty repetitions are sufficient to accurately characterize both the sparse and dense scenarios. The overall performance of the sparse and dense matrices shows only a weak dependence on the oversampling parameter p, suggesting that p primarily contributes to the robustness of the method rather than directly influencing the accuracy of the computed results. Nevertheless, some sensitivity to p can be observed in specific cases —for instance, in the matrix *steam3* at low values of *rank/n*. In such scenarios, where p is on the order of the target rank, the truncated SVD appears more susceptible to oversampling effects, which can noticeably impact the resulting metrics. However, it is important to emphasize that the general insensitivity to p implies that the performance curves shown in Figure 2 for small values of p remain valid and informative for characterizing matrix behavior. The observed increase in execution time with growing $rank/n$ (see bottom row of Figures 4 and 5) is attributed to the rising computational cost associated with the truncated SVD. In the maps corresponding to dense matrices (Figure 5), the iteration-to-solution metric exhibits a trend toward uniformity across the full $rank/n$ range. Given that all four dense matrices have exponentially decaying singular values —and are among those requiring the highest number of iterations to reach convergence— this relative uniformity suggests that employing a very low $rank/n$ in constructing the GMRES preconditioner may be rather advantageous. The performance maps indicate that only minimal spectral information is required to achieve excellent convergence.

6 RELATED WORK

Several authors have analyzed the performance of the GMRES and randomized-SVD algorithms independently. Research on GMRES is extensive, while publications on rSVD are more limited and recent. In

Table 2: Iteration-to-solution performance for sparse matrices. Deterministic SVD and rSVD formulations are compared for $rank/n = 0.02$ and 0.3 (2% and 30% of the leading singular triplets, respectively. The SVD-based ILU(0) preconditioning is also provided (rightmost column).

	SVD Preconditioning				ILU(0)
	$rank/n = 2\%$		$rank/n = 30\%$		
Matrix	SVD	rSVD	SVD	rSVD	
dw256B	4	4	3	3	4
pde225	18	21	11	12	19
cdde4	16	19	8	9	18
poisson2D	23	38	9	10	41
ex37	5	6	3	3	7
bfwa62	22	25	9	10	30
bwm200	20	31	5	6	38
ex1	35	36	12	13	32
ex22	80	131	13	14	196
orsirr_2	26	52	5	6	56
tub100	10	11	4	5	12
olm100	40	51	14	16	59
pores_1	10	10	4	4	10
steam1	5	4	2	2	5
DK01R	14	15	6	7	17
saylr1	15	19	5	6	19
steam3	5	5	1	1	6
lung1	24	24	23	23	23

Table 3: Iteration-to-solution performance for a subset of the dense matrix population. Deterministic SVD and rSVD formulations are compared for $rank/n = 0.02$ and 0.3 (2% and 30% of the leading singular triplets, respectively. The ILU(0)-based preconditioning is shown (rightmost column).

	SVD Preconditioning				ILU(0)
	$rank/n = 2\%$		$rank/n = 30\%$		
Matrix	SVD	rSVD	SVD	rSVD	
n250_exp_cond1e8	18	14	6	13	22
n1000_exp_cond1e8	22	14	22	15	22
n4000_exp_cond1e5	5	4	5	4	21
n4000_exp_cond1e8	23	21	23	21	23

the context of preconditioned restarted GMRES implementations, relevant studies have emerged focusing on GPU executions to accelerate the time-to-solution (Li and Saad 2013; Aliaga et al. 2018; Aliaga et al. 2017) with speedups up to ∼6x. In DeVries et al. (2013), a hybrid GPU-CPU version of Flexible-GMRES is implemented and its performance evaluated. It outperforms its serial CPU-only counterpart by a ∼22x speedup with the largest matrices tested.

Minimizing GMRES time-to-solution has been a focus for over two decades. Interestingly, some authors have explored the usage of a truncated-SVD to build a preconditioner for ill-posed problems (Eldén and Simoncini 2012) to improve the GMRES convergence. However, it is clear that for such an approach to be effective, an efficient implementation of the SVD algorithm is crucial, as it would otherwise become a time-consuming bottleneck. Thus, a randomized version of the SVD algorithm is a more economical choice

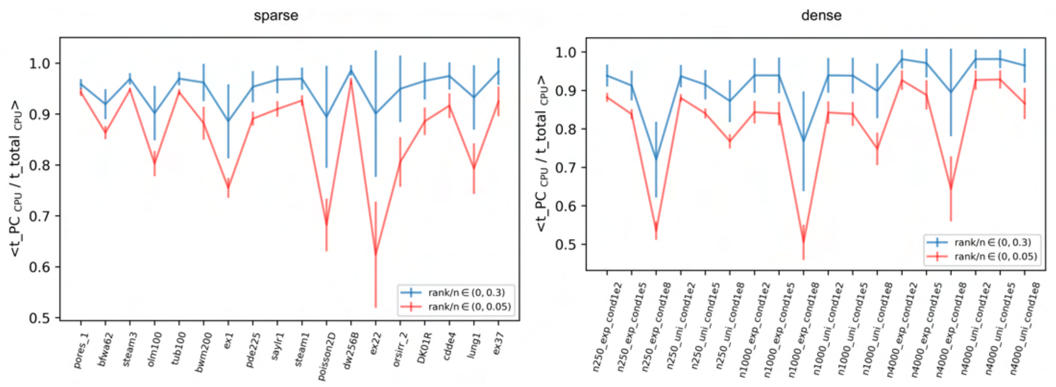


Figure 3: Ratio of the preconditioner (PC) building time and total execution time of the rSVD-based preconditioned GMRES, applied to the sparse (left column) and dense (right column) matrix sets. Mean value and range of variation are plotted. Matrices are sorted by increasing size.

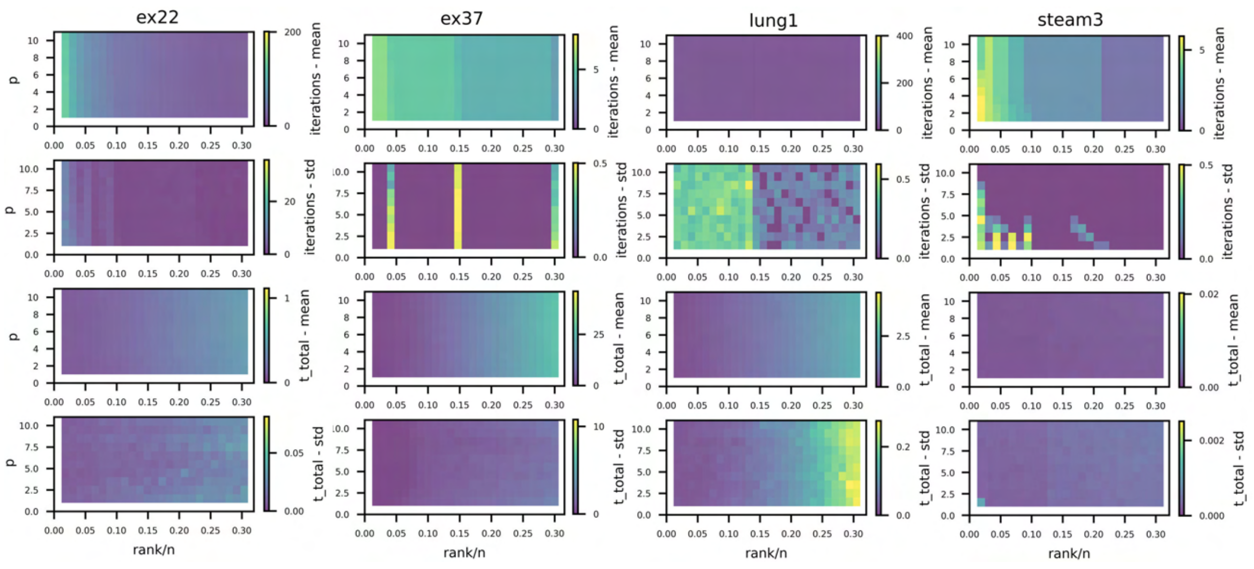


Figure 4: Performance maps of 4 sparse matrices: *ex22, ex37, lung1, steam3* set by columns. Rows correspond to the metrics: iteration-to-solution (*iterations - mean value*), its standard deviation (*iterations - std*), time-to-solution (*t_total - mean value*), and its standard deviation (*t_total - std*).

and is therefore closely linked to achieving good performance with a GMRES solver. An implementation for GPU is given in (Struski et al. 2024) and tested with synthetic dense matrices of prescribed spectra. Their GPU-based randomized SVD is now part of an official CUDA release. A promising approach involves the GPU-CPU hybridization of the rSVD code, as demonstrated in (Ji and Li 2014), where a hybrid GPU-CPU accelerated randomized-SVD is described. Specifically, the paper analyzes the contribution of each algorithmic block to the total computational cost and evaluates different implementation strategies using established linear algebra GPU libraries. It reports a speedup of ∼6x to ∼7x when applied to large random and image matrices.

The present investigation revisits the idea of a novel GMRES preconditioner described in (Higham and Mary 2019), which exploits a truncated rSVD performed on an error matrix derived from a low-accuracy *LU* factorization of matrix *A*, which is typically ill-conditioned. The randomized SVD captures the dominant

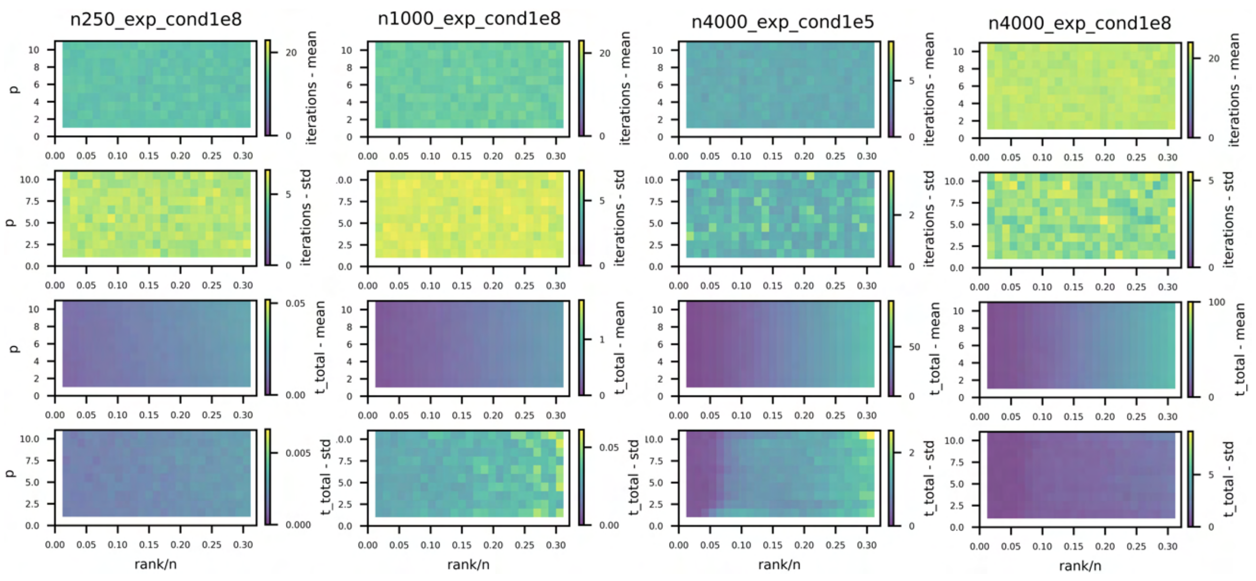


Figure 5: Performance maps of 4 dense matrices: *n250_exp_1e8, n1000_exp_1e8, n4000_exp_1e5, n4000_exp_1e8* set by columns. Rows correspond to the metrics: iteration-to-solution (*iterations - mean value*), its standard deviation (*iterations - std*), time-to-solution (*t_total - mean value*), and its standard deviation (*t_total - std*).

Table 4: Time-to-solution ratio $t_{rSVD}/t_{ILU(0)}$ of preconditioned GMRES using rSVD and ILU(0). Only the subset of dense matrices with the largest iteration count is included.

	$t_{rSVD}/t_{ILU(0)}$			$t_{rSVD}/t_{ILU(0)}$	
Matrix	min	max	Matrix	min	max
dw256B	0.83	1.07	olm100	0.28	0.89
pde225	0.63	1.11	pores_1	0.44	1.02
cdde4	0.51	1.13	steam1	0.42	0.86
poisson2d	0.24	0.92	DK01R	0.44	0.94
ex37	0.42	0.86	saylr1	0.33	1.03
bfwa62	0.33	0.85	steam3	0.36	1.72
bwm200	0.18	0.90	lung1	1.01	1.09
ex1	0.43	1.21	n250_k2_exp	0.66	0.67
ex22	0.08	0.71	n1000_k8_exp	0.67	0.72
orsirr_2	0.11	0.94	n4000_k5_exp	0.20	0.26
tub100	0.43	0.92	n4000_k8_exp	0.93	0.94

spectral content of matrix A, and the resulting low-rank approximation effectively accelerates GMRES. The authors prototype this randomized-SVD-based GMRES version in Matlab and compare the performance of both direct rSVD and via-ID rSVD approaches, conducting tests with a set of small matrices from the Suite Sparse Matrix Collection (Davis and Hu 2011). Additionally, the rSVD-based preconditioner is analyzed in (Moríñigo et al. 2022) from a resilience standview. The authors show that it improves the fault-tolerance behaviour compared to its ILU(0)-preconditioned counterpart. Moreover, it exhibits better resilience than FGMRES in scenarios where fault tolerance is critical, which is essential for large-scale parallel computing.

To the authors' knowledge, none of the previous research explores the performance of the rSVD-based GMRES within the context of a non-interpreted programming language. In this study, the algorithm is implemented in C language to investigate its performance in greater detail using a collection of 36 matrices.

7 CONCLUSIONS

The restarted GMRES algorithm with rSVD-based preconditioning has been implemented in C and tested on a set of 18 sparse and 18 dense matrices with a wide range of spectral properties. The results demonstrate that convergence can be achieved in many cases with fewer iterations than with ILU(0) by retaining only a small portion of the spectral content of the error matrix proposed as part of the present methodology. A crucial aspect is that a truncated SVD underlies in the rSVD algorithm, hence the cost of constructing the preconditioner becomes more computationally expensive as the chosen *rank* increases. Hence, for practical implementations it is necessary the condition $rank \ll n$ to be satisfied.

Furthermore, the computational overhead associated with the construction of the preconditioner in this serial implementation is significant compared to the overall execution time of the solver. This underscores the importance of developing a more efficient rSVD formulation which exploits the decomposition factors of the truncated error matrix derived from dense operations.

Lastly, it is stressed that this study focuses on serial implementations of rSVD into the preconditioned GMRES and does not address specific issues of parallel implementations, where inter-process communication can significantly affect performance. In large-scale parallel environments, where communication costs are non-negligible, the development of an efficient, scalable rSVD, particularly with communication-avoiding properties, is essential. Future research will analyze a parallel version of it applied to larger matrices in scenarios requiring a high number of iterations when using ILU(0), to better understand the behavior of this novel approach at scale.

ACKNOWLEDGMENTS

This work was partially funded by the EuroHPC-funded project EoCoE3 (Grant ID: 101144014), co-funded by the Spanish Ministry of Science, Innovation, and Universities MINERIA project (Grant ID: PID2024-157169OB-I00), and led within the SciTrack group tasks towards exascale computing. The authors thank the CETA-CIEMAT support team for access to their computing facilities.

REFERENCES

Agullo, E., M. Altenbernd, H. Anzt, L. Bautista-Gomez, T. Benacchio, L. Bonaventura, *et al*. 2021. "Resiliency in Numerical Algorithm Design for Extreme Scale Simulations". *International Journal of High Performance Computing Applications* 36(2):251–285 https://doi.org/10.1177/10943420211055188.

Aliaga, J. I., M. Bollhöfer, E. Dufrechu, P. Ezzatti, and E. S. Quintana-Ortí. 2017. "A Data-Parallel ILUPACK for Sparse General and Symmetric Indefinite Linear Systems". In *Prodeedings of the Euro-Par Workshops*, 121–133. Cham: Springer International Publishing https://doi.org/10.1007/978-3-319-58943-5_10.

Aliaga, J. I., E. Dufrechu, P. Ezzatti, and E. S. Quintana-Ortí. 2018. "An Efficient GPU Version of the Preconditioned GMRES Method". *The Journal of Supercomputing* 75:1455–1469 https://doi.org/10.1007/s11227-018-2658-1.

Benzi, M. 2002. "Preconditioning Techniques for Large Linear Systems: A Survey". *Journal of Computational Physics* 182(2):418–477 https://doi.org/https://doi.org/10.1006/jcph.2002.7176.

Davis, T. A., and Y. Hu. 2011. "The University of Florida Sparse Matrix Collection". *ACM Transactions on Mathematical Software* 38(1) https://doi.org/10.1145/2049662.

de Castro-Sánchez, M., J. A. Moríñigo, F. Terragni, and R. Mayo-García. 2025. "Analysis of the SVD Scaling on Large Sparse Matrices". In *Proceedings of the Winter Simulation Conference*, WSC '24, 2523–2534 https://doi.org/10.1109/WSC63780.2024.10838971.

DeVries, B., J. Iannelli, C. Trefftz, K. A. O'Hearn, and G. Wolffe. 2013. "Parallel Implementations of FGMRES for Solving Large, Sparse Non-symmetric Linear Systems". *Procedia Computer Science* 18:491–500.

Eldén, L., and V. Simoncini. 2012. “Solving Ill-Posed Linear Systems with GMRES and a Singular Preconditioner”. *SIAM Journal on Matrix Analysis and Applications* 33(4):1369–1394 https://doi.org/10.1137/110832793.

Ferrero-Roza, P., J. A. Moríñigo, and F. Terragni. 2024. “Strong Scaling of the SVD Algorithm for HPC Science: A PETSc-Based Approach”. In *Proceedings of the Winter Simulation Conference*, WSC ’23, 2872–2883 https://doi.org/10.1109/WSC60868.2023.10407904.

Halko, N., P.-G. Martinsson, and J. A. Tropp. 2009. “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions”. *arXiv preprint arXiv: 0909.4061*.

Henderson, H. V., and S. R. Searle. 1981. “On Deriving the Inverse of a Sum of Matrices”. *SIAM Review* 23(1):53–60.

Higham, N. J., and T. Mary. 2019. “A New Preconditioner that Exploits Low-Rank Approximations to Factorization Error”. *SIAM J. Sci. Comput.* 41(1):A59–A82 https://doi.org/10.1137/18M1182802.

Ji, H., and Y. Li. 2014. “GPU Accelerated Randomized Singular Value Decomposition and Its Application in Image Compression”. In *Proceedings of Modeling, Simulation, and Visualization Student Capstone Conference (MSVSCC 2014)* https://doi.org/10.25776/zrj4-5c48.

Li, R., and Y. Saad. 2013. “GPU-Accelerated Preconditioned Iterative Linear Solvers”. *Journal of Supercomputing* 63(2):443–466.

Martinsson, P. 2019. *Randomized methods for matrix computations*, Volume 25 of *IAS/Park City Mathematics Series*, 187–230.

Moríñigo, J. A., A. Bustos, and R. Mayo-García. 2022. “Error Resilience of Three GMRES Implementations under Fault Injection”. *J. Supercomput.* 78(5):7158–7185 https://doi.org/10.1007/s11227-021-04148-x.

Saad, Y. 1993. “A Flexible Inner-Outer Preconditioned GMRES Algorithm”. *SIAM Journal on Scientific Computing* 14(2):461–469.

Saad, Y. 2020. “Iterative Methods for Linear Systems of Equations: A Brief Historical Journey”. In *75 Years of Mathematics of Computation*, edited by S. C. Brenner, I. E. Shparlinski, C. Shu, and D. B. Szyld, Volume 754 of *Contemporary Mathematics*. American Mathematical Society.

Saad, Y., and M. H. Schultz. 1986. “GMRES: A Generalized Minimal RESidual Algorithm for Solving Nonsymmetric Linear Systems”. *SIAM Journal on Scientific and Statistical Computing* 7(3):856–869 https://doi.org/10.1137/0907058.

Saad, Y., and H. A. van der Vorst. 2000. “Iterative Solution of Linear Systems in the 20th Century”. *Journal of Computational and Applied Mathematics* 123(1):1–33 https://doi.org/https://doi.org/10.1016/S0377-0427(00)00412-X.

Struski, L., P. Morkisz, P. Spurek, S. R. Bernabeu, and T. Trzciński. 2024. “Efficient GPU Implementation of Randomized SVD and its Applications”. *Expert Systems with Applications* 248:123462 https://doi.org/10.1016/j.eswa.2024.123462.

van der Vorst, H. A. 2003. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press.

Vuik, C. 1995. “New Insights in GMRES-like Methods with Variable Preconditioners”. *Journal of Computational and Applied Mathematics* 61(2):189–204 https://doi.org/https://doi.org/10.1016/0377-0427(94)00067-B.

Vuik, C. 2018. “Krylov Subspace Solvers and Preconditioners”. *ESAIM: ProcS* 63:1–43 https://doi.org/10.1051/proc/201863001.

AUTHOR BIOGRAPHIES

JOSÉ A. MORÍÑIGO is a senior researcher at the Department of Technology of the Centre for Energy, Environmental and Technological Research (CIEMAT). He earned his PhD in Aeronautical Engineering from Universidad Politécnica de Madrid (2004). His research interests include the development of scalable, resilient solvers of PDEs and numerical algebra for supercomputers and their application to fluid dynamics. He has a long experience in simulation of turbulent compressible flow, mostly applied to aerospace propulsion. He is a lecturer of space propulsion at the School of Industrial Engineering of Bilbao. His e-mail is josea.morinigo@ciemat.es and his research group website is http://rdgroups.ciemat.es/web/sci-track.

ANDRÉS BUSTOS is a senior researcher at the Department of Technology of the Centre for Energy, Environmental and Technological Research (CIEMAT). He earned his PhD in Physics from Universidad Complutense de Madrid (2012). His areas of work include numerical simulation of fusion plasma transport, scientific computing and applications of artificial intelligence. His e-mail is andres.bustos@ciemat.es and his research group website is http://rdgroups.ciemat.es/web/sci-track.

RAFAEL MAYO-GARCÍA is a senior researcher at CIEMAT, Harvard University Fellow, and coordinator of the European EERA Joint Programme ’Digitalisation for Energy’. He earned his PhD in Physics from Universidad Complutense de Madrid (2004). He has been involved in many experiments in the US, Bulgaria, Sweden, and Ireland (funded, among others, by the European Commission with a Marie Curie Action). He has also obtained a postdoctoral fellowship in the Spanish Juan de la Cierva Programme. He authored more than 160 scientific articles. He has participated in 64 projects (being PI in 9 out of them) and has been involved in several European and National initiatives working on HPC & BD scientific developments. He also has served several institutions as an evaluator for their competitive Calls, European Commission included and has supervised 8 theses. His email is rafael.mayo@ciemat.es.