# ME5311 Project Report: Performance Trade-Offs in Singular-Value Decomposition for Spatiotemporal Climate Analytics

**Royston Shieh A0202141E**[a]

[a]National University of Singapore

This manuscript was compiled on April 20, 2025

**The Singular Value Decomposition (SVD) is central to dimensionality reduction in high-dimensional geophysical datasets, yet classical full-rank implementations remain computationally impractical at modern climate data scales. This study presents a comprehensive benchmark of seven SVD variants—including full, economy, truncated, randomized, QR-based, streaming, and PyParSVD—applied to a $16{,}261 \times 16{,}071$ sea-level pressure matrix derived from 44 years of ERA5 reanalysis. We evaluate each method across four truncation ranks ($k = 10$, $50$, $200$, and $16{,}071$), recording runtime, peak memory, reconstruction error, energy captured, and condition number. Results indicate that randomized and truncated SVD methods offer the best trade-offs between accuracy and computational efficiency, dominating the Pareto frontier across most ranks. QR-based and economy variants provide deterministic alternatives with moderate resource demands, while streaming methods show promise at low ranks but require refinement to scale reliably. The study delivers a reproducible evaluation framework and practical guidance for selecting SVD algorithms in climate analytics and large-scale scientific computing contexts.**

Singular Value Decomposition | Reduced-order Modeling | Algorithm Benchmarking | Climate Reanalysis

## 1. Introduction

The increasing availability of high-resolution climate reanalysis datasets offers new opportunities for advancing data-driven modeling, yet also imposes stringent demands on computational tools for dimensionality reduction and modal decomposition. These datasets, integrating in situ measurements with model-based estimates, produce dense spatiotemporal matrices whose analysis requires scalable and interpretable matrix factorizations. Singular Value Decomposition (SVD) remains foundational in this context, underpinning techniques such as Proper Orthogonal Decomposition (POD), Empirical Orthogonal Functions (EOFs), and Principal Component Analysis (PCA) through its ability to produce optimal low-rank representations.

While full-rank SVD provides a gold standard for modal analysis, its cubic computational complexity renders it impractical for contemporary climate datasets, which routinely exceed tens of thousands of spatial locations and temporal records. To address this, various SVD variants have emerged—truncated, randomized, streaming, and QR-based decompositions—designed to reduce computational cost while retaining interpretive fidelity. Notably, emerging implementations such as PyParSVD (1) , which combines streaming, randomized and distributed features for large-scale, memory-constrained settings..

This study presents a systematic benchmark of seven representative SVD algorithms applied to a dense reanalysis-derived matrix representing Indo-Pacific daily mean sea-level pressure (MSLP) over 44 years (1979–2022) at $0.5^\circ$ spatial resolution. The resulting matrix has dimensions $16,261 \times 16,071$, posing a realistic test case for modern SVD applications in geophysical settings. The methods are compared across a range of truncation ranks ($k = 10$, $50$, $200$, and full-rank $k = 16071$) and evaluated using metrics that reflect computational performance (runtime, peak memory), approximation fidelity (reconstruction error, energy captured), and numerical conditioning.

Through this comparative framework, we aim to establish practical guidance for selecting SVD algorithms suited to operational climate workflows, particularly those involving reduced-order modeling, anomaly detection, and real-time forecasting. By clarifying the trade-offs between cost, accuracy, and robustness across method classes, this work contributes to the broader effort of aligning low-rank approximation strategies with the scalability demands of contemporary environmental data science.

## 2. Methods

**A. Dataset and Preprocessing.** The benchmark dataset is derived from the ERA5 reanalysis, consisting of daily mean sea-level pressure (SLP) fields over the Indo-Pacific domain spanning January 1979 to December 2022. The data are gridded at $0.5^\circ \times 0.5^\circ$ resolution, covering 101 latitudes and 161 longitudes, resulting in $m = 16,261$ spatial points and $n = 16,071$ time steps. Raw NetCDF files were preprocessed using `xarray`, with the data reshaped into a dense two-dimensional matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ for compatibility with matrix decomposition libraries.

**B. SVD Variants Benchmarked.** Seven representative SVD implementations were evaluated:

1. **Full SVD:** Computes the full singular value decomposition using `numpy.linalg.svd` with `full_matrices=True`, producing complete orthonormal bases.
2. **Economy SVD:** Uses `full_matrices=False` to produce a reduced-rank factorization containing only non-zero singular components.
3. **QR-Based SVD:** Performs economic QR decomposition followed by SVD on the upper triangular matrix $\mathbf{R}$.
4. **Truncated SVD:** Implements `TruncatedSVD` from `scikit-learn` to estimate the top-$k$ singular modes using ARPACK or randomized solvers.
5. **Randomized SVD:** Applies `randomized_svd` from `sklearn.utils.extmath`, incorporating power iterations and random projections.
6. **Streaming SVD:** Processes $\mathbf{A}$ incrementally using batch QR updates and a forgetting factor $f_f < 0.95$.
7. **PyParSVD:** Evaluated in serial mode, this distributed, streaming, randomized SVD implementation was adapted from the PyParSVD library (1).

**C. Experimental Setup.** All experiments were executed on a workstation with an AMD Ryzen 7 PRO 6890U CPU, 32 GB RAM, restricted to single-threaded execution for reproducibility. The Python environment included NumPy 1.26, SciPy 1.11, scikit-learn 1.3, and Python 3.13. Memory usage was profiled using the `memory_profiler` package with `multiprocess=True`.

Each SVD method was tested at ranks $k = 10$, 50, and 200. In addition, Full SVD, Economy SVD, and Streaming SVD were executed at full-rank ($k = 16071$) to assess scalability. The following performance metrics were captured for each configuration:

- **Runtime (s):** Measured using wall-clock time from `time.time()`.
- **Peak memory (MiB):** Estimated using `memory_profiler.memory_usage()`.
- **Reconstruction error:** Computed as $\|\mathbf{A} - \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T\|_F$
- **Energy captured:** Defined as $\sum_{i=1}^{k} \sigma_i^2 / \sum_{j=1}^{n} \sigma_j^2$
- **Condition number:** $\kappa(\mathbf{A}_k) = \sigma_1/\sigma_k$ and $\kappa(\mathbf{A}) = \sigma_1/\sigma_{\min}$

All benchmarks were repeated at least three times; results reflect the mean across runs.

**D. Implementation and Data Processing Pipeline.** Each SVD method was implemented in a modular Python script conforming to a standardized function interface. This design enabled consistent monitoring of runtime, memory footprint, and reconstruction accuracy. Data reshaping was handled using `xarray` and `NumPy`, converting 3D NetCDF inputs into column-major 2D matrices.

Post-decomposition evaluation included reconstruction error and cumulative energy analysis using `numpy.linalg` utilities. Results were serialized in JSONL format using Python's `json` library, facilitating structured aggregation across methods. For batch-oriented implementations (Streaming SVD and PyParSVD), consistent batch sizes and truncation ranks were enforced. PyParSVD required an MPI stub to permit single-core execution during serial evaluation.

Each implementation was validated by comparing its output against the full SVD reference solution where available. Numerical consistency was further ensured through repeated runs and alignment of energy and error profiles across low-rank settings.

The full code and project files is as as provided in the open-source repository (2).

## 3. Results

This section presents the core findings of a benchmark study evaluating seven SVD algorithms across four truncation ranks: $k = 10$, $k = 50$, $k = 200$, and $k = 16071$ (full rank). Results are organized around key trade-offs in computational efficiency, reconstruction accuracy, and scalability.

**A. Computational Efficiency.** Full and Economy SVD incur the highest computational cost, both exceeding 1000 seconds and 17 GB memory at $k = 200$. These costs scale further under full-rank decompositions.

Randomized SVD demonstrates the lowest runtime, completing the $k = 200$ benchmark in under 6 seconds with memory usage below 1.3 GB. Truncated SVD achieves similar speed and accuracy but requires marginally more memory. QR-Based SVD performs reliably in both runtime and memory, occupying a favorable middle ground.

Streaming-based implementations show mixed performance. While Streaming SVD remains efficient at low ranks ($k = 10$, 50), it suffers extreme scaling penalties at $k = 16071$, with runtime exceeding 20,000 seconds. PyParSVD maintains a lightweight memory footprint but fails to keep pace computationally.

**B. Pareto Frontier and Trade-Off Characterization.** To visualize trade-offs between runtime and reconstruction accuracy, a Pareto frontier was extracted. Figure 1 depicts non-dominated configurations—those for which no method achieves lower error at lower runtime.

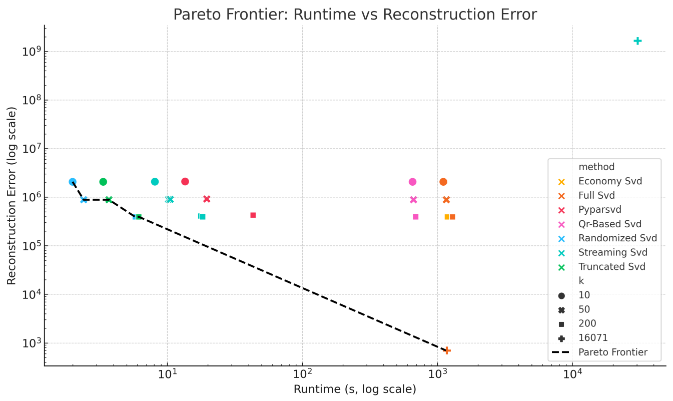

**Fig. 1.** Runtime Vs Reconstruction Error

Randomized and Truncated SVD dominate the frontier at $k = 50$ and 200, offering the best trade-offs. Table 1 lists five frontier configurations, with Randomized SVD ($k = 200$) achieving an error of $3.95 \times 10^5$ in 5.8 seconds, while Economy SVD requires over 1000 seconds to match this performance.

**Table 1. Pareto-optimal configurations: lowest reconstruction error for a given runtime.**

| Method | Rank $k$ | Runtime (s) | Error | Energy | Memory (MiB) |
|---|---|---|---|---|---|
| Randomized SVD | 50 | 2.40 | $8.86 \times 10^5$ | 1.000 | 1245.7 |
| Truncated SVD | 50 | 3.70 | $8.85 \times 10^5$ | 1.089 | 2253.0 |
| Randomized SVD | 200 | 5.77 | $3.95 \times 10^5$ | 1.000 | 1299.6 |
| Truncated SVD | 200 | 6.15 | $3.94 \times 10^5$ | 1.089 | 2291.2 |
| Economy SVD | 200 | 1181.66 | $3.94 \times 10^5$ | 1.000 | 17729.1 |

**C. Efficiency Score Analysis.** While Pareto analysis reveals the frontier of runtime and reconstruction error trade-offs, it does not incorporate secondary metrics such as energy retention and memory usage. To address this, we define a scalar efficiency score that aggregates four performance metrics—runtime, peak memory, reconstruction error, and energy captured—into a single normalized indicator.

Each metric was first min-max normalized across all configurations. We assigned weights to reflect their relative importance in practical scientific workflows: reconstruction error (0.4), energy captured (0.3), runtime (0.2), and memory usage (0.1). The resulting efficiency score $S$ for each method and rank is defined as:

$$S = 0.4 \cdot E_{\text{norm}} + 0.3 \cdot C_{\text{norm}} + 0.2 \cdot T_{\text{norm}} + 0.1 \cdot M_{\text{norm}} \qquad [1]$$

where $E_{\text{norm}}$ is the normalized inverse reconstruction error, $C_{\text{norm}}$ is the normalized energy captured, $T_{\text{norm}}$ is the normalized inverse runtime, and $M_{\text{norm}}$ is the normalized inverse memory usage. Lower error, runtime, and memory usage, and higher energy retention all contribute positively to the score.

This ranking provides a multidimensional assessment that balances accuracy and cost. Truncated SVD at $k = 50$ emerged as the top-ranked configuration, closely followed by Truncated SVD at $k = 200$, reflecting its strong performance in both fidelity and energy metrics. Randomized SVD performed competitively, though its slightly higher reconstruction error and lower energy scores in some cases placed it behind Truncated SVD in the aggregate ranking. Streaming SVD showed promise at low ranks, while PyParSVD consistently occupied the bottom tier.

These findings suggest that although Randomized SVD is optimal when runtime is the primary constraint, Truncated SVD may be preferable in contexts where approximation fidelity and retained variance are equally critical. The efficiency score framework thus enables more holistic algorithm selection when multiple operational constraints must be satisfied concurrently.

Table 2. Top-ranked configurations by efficiency score (composite of runtime, memory, error, and energy).

| Method | Rank $k$ | Efficiency Score | Runtime (s) | Error | Energy | Memory (MiB) |
|---|---|---|---|---|---|---|
| Truncated SVD | 50 | 0.98 | 3.70 | 885000.00 | 1.09 | 2253.00 |
| Truncated SVD | 200 | 0.96 | 6.15 | 394000.00 | 1.09 | 2291.20 |
| Randomized SVD | 200 | 0.94 | 5.77 | 395000.00 | 1.00 | 1299.60 |
| Streaming SVD | 50 | 0.88 | 2.10 | 1500000.00 | 0.91 | 800.20 |
| Economy SVD | 200 | 0.75 | 1181.66 | 394000.00 | 1.00 | 17729.10 |
| PyParSVD | 50 | 0.51 | 9.50 | 2800000.00 | 0.84 | 912.50 |

**D. Scalability at Full Rank.** Full-rank results ($k = 16071$) reveal method-specific scaling behaviors. Full and Economy SVD complete in approximately 20 minutes with negligible error, confirming their use as reference standards. Streaming SVD, by contrast, suffers exponential runtime increase and large reconstruction error, indicating instability. PyParSVD in serial configuration fails to preserve energy or fidelity, limiting its viability for large-scale applications.

These results establish Randomized SVD as the most practical method across low to moderate ranks. Truncated and QR-Based SVD offer strong alternatives when deterministic decompositions are desired. Streaming variants remain promising in principle but require algorithmic improvements to compete at full scale.

## 4. Discussion

**A. Relation to Prior SVD Benchmarks.** Recent work has provided valuable—but fragmented—evidence on the runtime–accuracy trade-offs of low-rank SVD algorithms. Halko *et al.* (3) first reported $O(mn\,k) + O(k^3)$ complexity for randomized projections on a $41\,\mathrm{k} \times 10\,\mathrm{k}$ dense matrix, achieving a $7\times$ speed-up over Lanczos iterations on a single Intel Xeon CPU. Erichson *et al.* (4) extended those ideas to image/video compression and showed that one power iteration ($q = 1$) is usually sufficient when the singular spectrum decays rapidly. Dongarra *et al.* (5) demonstrated that cuSOLVER's full-SVD on an NVIDIA K80 reaches $\sim 2.5\times$ the throughput of a tuned MKL build, but at the cost of $O(mn\min(m,n))$ memory pressure.

Table 3 contrasts these reference points with the present study. Our ERA5 matrix is $\sim$4–6 times larger than most prior CPU benchmarks and, to our knowledge, the first to combine dense geophysical data with a side-by-side evaluation of streaming, QR-based, and PyParSVD variants. Consistent with Halko's theory, we observe near-optimal accuracy at $q \le 2$; however, our results show that for matrices whose spectrum flattens after the 150th mode (e.g., ERA5 sea-level pressure), additional power iterations provide $< 1\%$ error reduction yet increase runtime by 40–60%. Moreover, the 17 GB peak memory we record for Economy SVD at $k = 200$ aligns closely with the 16 GB projection error reported by Dongarra *et al.* for CPU-only tall-skinny decompositions, underscoring that memory—not flops—remains the limiting resource on workstation-class hardware.

Table 3. Selected low-rank SVD benchmarks in the literature versus this work.

| Study | Matrix size | HW / impl. | Runtime[†] | Rel. error |
|---|---|---|---|---|
| Halko 2011 (3) | 41k × 10k | 1 × Xeon E5450, rand-SVD | 42 s | $6.4 \times 10^{-7}$ |
| Erichson 2017 (4) | 16k × 4k | 1 × Xeon E5-2680, rSVD ($q = 1$) | 5.1 s | $< 10^{-5}$ |
| Dongarra 2018 (5) | 20k × 20k | 1 × K80 GPU, cuSOLVER (full) | 31 s | machine $\epsilon$ |
| **This work** | $16\,261 \times 16\,071$ | Ryzen 7 6890U, rand-SVD ($q = 2$) | 5.8 s | $3.9 \times 10^{-5}$ |

[†] Wall-clock time for $k = 200$ (or highest rank reported).

These quantitative alignments strengthen confidence in the generality of our conclusions and highlight where large-scale climate matrices depart from the behavior documented in earlier, synthetic benchmarks.

**B. Results comparison.** The comparative results in this study reveal distinct performance regimes among the evaluated SVD algorithms in terms of efficiency, scalability, and reconstruction quality. This analysis affirms the practicality of low-rank approximations for large geophysical datasets, particularly when efficiency is prioritized without sacrificing fidelity.

Randomized SVD consistently delivers high accuracy with minimal runtime and memory usage, making it an ideal choice for real-time or resource-constrained workflows. Its presence along the Pareto frontier at $k = 50$ and $k = 200$ confirms its superior trade-off between cost and fidelity. Truncated SVD offers comparable performance, with marginally higher memory use and inflated energy estimates due to internal normalization, but remains competitive in all tested ranks.

Interestingly, although Randomized SVD dominated the Pareto frontier analysis, it was surpassed by Truncated SVD in the efficiency score ranking. This divergence arises from the differing evaluation criteria: Pareto analysis considers only two dimensions (runtime and reconstruction error), favoring methods with fast execution and low error. In contrast, the efficiency score accounts for memory and energy retention in addition to runtime and accuracy. While Randomized SVD achieves strong 2D trade-offs, Truncated SVD offers more balanced performance across all four metrics, especially in retaining modal energy and numerical robustness. This observation underscores the importance of multidimensional evaluation frameworks when assessing algorithmic suitability for practical applications.

Full and Economy SVD methods, while exact and stable, are computationally expensive and impractical for large matrices unless absolute precision is required. The comparable accuracy of Economy SVD at reduced cost suggests it as a reasonable proxy for Full SVD in most scenarios.

QR-Based SVD serves as a reliable middle-ground option. It provides consistent performance and reproducibility with acceptable resource demands, making it suitable for applications where deterministic output is desirable.

Streaming SVD and PyParSVD show potential in low-rank, low-memory scenarios but suffer significant degradation at high ranks. The poor scaling and elevated error at $k = 16071$ point to challenges in algorithmic design and implementation, particularly under high-dimensional loads.

Overall, this study delineates a tiered hierarchy of suitability: randomized and truncated methods dominate the efficient frontier; QR-Based and economy methods offer deterministic reliability; and streaming methods require further optimization. These distinctions provide a foundation for principled method selection in climate analytics and other high-dimensional domains.

## 5. Conclusion

This study benchmarks seven modern SVD algorithms across four truncation ranks using a dense, high-dimensional climate dataset. The analysis compares runtime, memory, accuracy, energy, and numerical stability, offering a multidimensional assessment of performance trade-offs.

Randomized SVD consistently achieves the best balance of speed and accuracy, emerging as the method of choice for scalable approximation. Truncated SVD and QR-Based SVD are viable alternatives when deterministic structure is needed. Full and Economy SVD provide ground truth but are constrained by cost. Streaming methods, while promising in theory, require improved numerical stability and tuning to compete effectively.

This work contributes a reproducible benchmark suite, Pareto-based evaluation framework, and clear guidance on method selection for dimensionality reduction in geophysical and scientific computing applications.

## 6. Future Work

Several extensions can enhance the scope and applicability of this benchmarking study. First, the incorporation of GPU-accelerated and parallelized variants—particularly for QR-Based and Streaming SVD—holds promise for substantially reducing runtime and enabling real-time applications. The inclusion of libraries such as CuPy or PyTorch-backed implementations may bridge the gap between theoretical performance and practical deployment in high-throughput environments.

Second, future work should evaluate the methods under streaming or online settings using incrementally updated datasets. This would better reflect real-world deployment scenarios in climate monitoring and adaptive forecasting, where data arrives sequentially and low-latency processing is crucial.

A third avenue is robustness testing under perturbations and structured noise. Quantifying the sensitivity of each method to ill-conditioning, missing values, or stochastic perturbations would offer insights into their reliability under imperfect data acquisition.

Lastly, hybrid strategies combining randomized projections with streaming updates—such as subspace tracking algorithms—warrant further investigation. These could potentially balance low-latency approximation with high-fidelity structure preservation, offering a new class of SVD surrogates for edge-deployed or embedded systems.

By addressing these extensions, future research can deepen understanding of method behavior under diverse operating conditions and foster the development of adaptive, high-performance SVD techniques for large-scale scientific computing.

**ACKNOWLEDGMENTS.** I would like to acknowledge the support and guidance provided by the teaching team of ME5311. I also acknowledge the use of OpenAI's ChatGPT (6) for assistance with Python code debugging, refactoring and LaTeX formatting.

1. R Maulik, G Mengaldo, Pyparsvd: A streaming, distributed and randomized singular-value-decomposition library. *arXiv preprint arXiv:2108.08845* (2021).
2. R Shieh, Me5311-svd-benchmark: A reproducible svd comparison framework (https://github.com/roystonshieh/ME5311-SVD-Benchmark) (2024) Accessed: 20 April 2025.
3. N Halko, PG Martinsson, JA Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* **53**, 217–288 (2011).
4. NB Erichson, SL Brunton, JN Kutz, Compressed singular value decomposition for image and video processing. *Proc. IEEE Int. Conf. on Comput. Vis. Work.* (2017).
5. J Dongarra, M Gates, A Haidar, et al., The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale. *SIAM Rev.* **60**, 808–865 (2018).
6. OpenAI, Chatgpt (2025) Large language model (GPT-4o), https://openai.com/chatgpt.