COMPUTING MATRIX EXPONENTIALS USING KRYLOV SUBSPACE METHOD

TINA ZHANG

Abstract. This paper investigates the computation of matrix exponentials using Krylov subspace methods, comparing their performance with classical approaches such as Taylor series expansions and direct methods. We present theoretical foundations, implementation details, and numerical experiments across different matrix types, including small dense matrices, symmetric negative definite matrices, and sparse Laplacian matrices. Our results demonstrate that Krylov subspace methods offer a robust balance between accuracy and computational efficiency, particularly for large-scale problems. The study provides practical recommendations for selecting appropriate methods based on matrix properties and problem requirements.

Keywords: matrix exponential, Krylov subspace, Arnoldi iteration, numerical linear algebra, sparse matrices

AMS subject classifications. 65F60, 65F50, 65L05

1. Introduction.

Matrix exponentials play a fundamental role in numerous scientific computing applications, from solving systems of ordinary differential equations to quantum mechanics and network analysis. The computation of $e^{tA}v$ for a matrix $A \in \mathbb{C}^{n\times n}$, scalar t, and vector $v \in \mathbb{C}^n$ presents significant numerical challenges, particularly for large-scale problems.

1.1. Basic Properties of Matrix Exponentials.

In this section, we discuss the fundamental properties of matrix exponentials, beginning with a formal definition.

DEFINITION 1 (Matrix Exponential). *Let A be an $n \times n$ real or complex matrix. The exponential of A, denoted by e^A or $\exp(A)$, is the $n \times n$ matrix given by the power series:*

$$e^A := \sum_{k=0}^{\infty} \frac{1}{k!} A^k.$$

In many applications, we are more interested in e^{tA}, where t is typically a small positive scalar, often used for time steps. Formally, we have:

$$e^{tA} = I + tA + \frac{t^2}{2!}A^2 + \cdots . \tag{1.1}$$

THEOREM 1.1. *The matrix series defined in (1.1) exists for all A for any fixed value of t, and for all t for any fixed A. It converges uniformly in any finite region of the complex t-plane.*

Let A and B be $n \times n$ complex matrices, and t and s be arbitrary complex numbers. Denote the identity matrix by I and the zero matrix by 0. The matrix exponential satisfies the following properties:

1. $e^0 = I$
2. $e^{(s+t)A} = e^{sA}e^{tA}$
3. $e^{tA}e^{-tA} = I$, so e^{tA} is never singular.
4. If A and B commute (i.e., $AB = BA$), then $e^{t(A+B)} = e^{tA}e^{tB}$.
5. If B is invertible, then $e^{B^{-1}AB} = B^{-1}e^AB$.
6. $\frac{d}{dt}e^{tA} = Ae^{tA}$
7. $\det(e^A) = e^{\mathrm{trace}(A)}$

For the classical problem of solving homogeneous systems of ordinary differential equations:

$$\dot{x}(t) = Ax(t),$$

with the initial condition $x(0) = x_0$, the solution is given by:

$$x(t) = e^{tA}x_0.$$

The computation of matrix exponentials, e^{tA}, is a fundamental problem in numerical linear algebra with wide-ranging applications in differential equations, quantum mechanics, control theory, and network analysis. For the linear system of ordinary differential equations:

$$\dot{x}(t) = Ax(t), \quad x(0) = x_0,$$

the solution is given by:

$$x(t) = e^{tA}x_0,$$

where e^{tA} is defined via the convergent power series:

$$e^{tA} = I + tA + \frac{t^2A^2}{2!} + \cdots.$$

1.2. Classical methods for computing matrix exponentials. Efficient and accurate computation of e^{tA} or its action on a vector $e^{tA}v$ is critical in both theory and practice. A variety of methods have been developed to compute matrix exponentials, each with distinct advantages and limitations. The seminal survey by Moler and Van Loan [5] (*"Nineteen Dubious Ways to Compute the Exponential of a Matrix"*) highlights the numerical pitfalls and trade-offs among classical approaches. For small, dense matrices, Padé approximation with scaling and squaring (as implemented in MATLAB's `expm`) is the gold standard. However, for large-scale sparse matrices—ubiquitous in modern scientific computing—these direct classical methods such as diagonalization, Taylor series, or Padé approximations become computationally infeasible for high-dimensional matrices due to $\mathcal{O}(n^3)$ complexity and memory constraints. We first introduce three classical methods to compute matrix exponentials: diagonalization, Taylor series, and Padé approximation before introducing the Krylov Subspace method.

1.2.1. Diagonalization Method. The diagonalization method is based on the fact that if a matrix A is diagonalizable, then it can be written as:

$$A = V\Lambda V^{-1}$$

where V is a matrix whose columns are the eigenvectors of A, and Λ is a diagonal matrix whose entries are the eigenvalues of A. Using this factorization, the matrix exponential e^{tA} can be computed as:

$$e^{tA} = Ve^{t\Lambda}V^{-1}$$

Since Λ is diagonal, the exponential of Λ is simply the exponential of each diagonal element:

$$e^{t\Lambda} = \mathrm{diag}(e^{t\lambda_1}, e^{t\lambda_2}, \ldots, e^{t\lambda_n})$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A. Therefore, the matrix exponential can be computed efficiently by first diagonalizing A, computing the exponentials of the eigenvalues, and then multiplying the result by V and V^{-1}.

This method is efficient when A is diagonalizable, but it may not work if A is defective (i.e., it cannot be diagonalized). In such cases, other methods, such as the Jordan canonical form or Krylov subspace methods, must be used.

1.2.2. Taylor Series Method. The Taylor series method is one of the most fundamental approaches for computing the matrix exponential e^A. This class of methods stems from the idea of approximating the scalar function e^z. These techniques treat the matrix exponential as a matrix function, analogous to the scalar exponential function. As a result, the matrix's specific properties, such as its order and eigenvalues, do not directly influence the computation.

The Taylor series method applies to the definition of e^A, given by the series:

$$e^A = \sum_{j=0}^{\infty} \frac{A^j}{j!} = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots \tag{1.2}$$

Let $P_K(A)$ denote the partial sum of the series (1.4). Liou [4] provides an error bound that can be used as a truncation criterion:

$$\|P_K(A) - e^A\| \leq \frac{\|A\|^{K+1}}{(K+1)!} \left(\frac{1}{1 - \frac{\|A\|}{K+2}} \right).$$

While this is the most fundamental method for computing the matrix exponential, it is not always satisfactory. Extreme examples, as presented in [5], illustrate its shortcomings, such as catastrophic cancellation and accuracy issues.

1.2.3. Padé Approximation Method. A Padé approximant is an approximation of a function by a rational function of a given order, such that the power series of the approximant agrees with the power series of the function it approximates. The (p, q) Padé approximation to e^A is defined as:

$$R_{pq}(A) = [D_{pq}(A)]^{-1} N_{pq}(A),$$

where

$$N_{pq}(A) = \sum_{j=0}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!} A^j$$

and

$$D_{pq}(A) = \sum_{j=0}^{q} \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!} (-A)^j.$$

The non-singularity of $D_{pq}(A)$ is guaranteed if p and q are sufficiently large. Diagonal approximants, where $p = q$, are typically preferred over off-diagonal approximants. To understand this, consider the case where $p < q$. In this case, the number of floating-point operations (flops) required to compute the off-diagonal approximant $R_{pq}(A)$ is proportional to qn^3, which is the same amount of work required to compute the diagonal approximant $R_{qq}(A)$ of higher order, where $2q > p+q$. Therefore, diagonal approximants can be expected to be more accurate for the same amount of computational effort.

2. Krylov Subspace Method. Although these classical methods provide essential ways to compute matrix exponentials, for large-scale sparse matrices—ubiquitous in modern scientific computing, they become computationally infeasible for high-dimensional matrices due to $\mathcal{O}(n^3)$ complexity and memory constraints.

The Krylov subspace method for matrix exponentials, introduced by Gallopoulos and Saad [6], provides an efficient approach for large-scale problems. The key idea is to project the matrix exponential onto a low-dimensional Krylov subspace, which significantly reduces computational cost while maintaining accuracy. This method draws from the success of Krylov methods in solving linear systems (e.g., the Conjugate Gradient method [7]) and eigenvalue problems (e.g., Arnoldi iteration [8]).

Krylov subspace methods are widely used in exponential integrators for stiff ODEs and PDEs, quantum dynamics for simulating time evolution in quantum systems, network analysis for computing centrality measures and diffusion processes, and control theory for solving linear dynamical systems.

Beyond the three classical alternative algorithms for computing matrix exponentials, there also exists Chebyshev Polynomial Expansion, which is efficient for Hermitian matrices but not general matrices; Lanczos Method, a variant of Krylov subspace methods for symmetric matrices; and rational approximations, which are best suited for specific matrix structures.

Variants of the Krylov method, such as Restarted Krylov Methods (Hochbruck & Lubich, 1997 [3]), Rational Krylov Methods for improved convergence (Güttel, 2013 [2]), and Adaptive Subspace Selection to dynamically adjust Krylov dimensions (Afanasjew et al., 2008 [1]), have been propoposed later to improve performance and accuracy.

2.1. Krylov Subspace and Arnoldi Algorithm. We formalize the Krylov Subspace method in this section. Given a matrix A, a vector v, and Krylov dimension m, the Krylov subspace is defined as:

$$\mathcal{K}_m(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{m-1}v\}.$$

An orthonormal basis V_m for the subspace is constructed using the Arnoldi iteration, which yields:

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T,$$

where H_m is an upper Hessenberg matrix.

The Arnoldi algorithm for Krylov subspaces proceeds as follows:

Algorithm 2.1 Arnoldi Iteration

```
v_1 ← v/‖v‖_2
for j = 1 to m do
    w ← A v_j
    for i = 1 to j do
        h_{i,j} ← w^* 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}
end for
```

This generates an orthonormal basis $\{v_1, v_2, \ldots, v_m\}$ for the Krylov subspace. The matrix $V_m := [v_1, v_2, \ldots, v_m]$ is orthogonal, and the upper Hessenberg matrix H_m satisfies $AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T$.

2.2. Projection and Approximation. The matrix exponential action is approximated as:

$$e^{tA} v \approx \|v\|_2 \, V_m e^{tH_m} e_1,$$

where $e_1 = (1, 0, \ldots, 0)^T$, and the small matrix exponential e^{tH_m} is computed via Padé approximation. The Arnoldi approximation to $e^A v$ is then given by:

$$e^A v \approx V_m e^{H_m} e_1.$$

The error bound for this approximation is given by:

$$\|e^A v - \beta V_m e^{H_m} e_1\|_2 \leq 2\beta \frac{\rho^m}{m!} \left(1 - \frac{\|A\|}{\rho^{m+2}}\right),$$

where $\rho = \|A - \alpha I\|_2$ for any scalar α, and $\beta = \|v\|_2$.

2.3. Key Properties and Computational Complexity.

(i) *Convergence*: The convergence rate depends on the eigenvalue distribution of A, and it is faster when the eigenvalues are clustered.

(ii) *Error Bound*: The error bound for the approximation is given by:

$$\left\| e^{tA}v - V_m e^{tH_m} e_1 \right\| \leq \frac{C(t\|A\|)^m}{m!}.$$

(iii) *Complexity*: The computational complexity of the Arnoldi algorithm is:

$$\mathcal{O}(m^3 + m \cdot \text{nnz}(A)),$$

where $\text{nnz}(A)$ is the number of nonzero elements in A.

2.4. Refined Error Bounds for Specific Matrix Types. For symmetric matrices, more refined error bounds have been derived. For example:

$$\epsilon_m \leq 10e^{-\frac{m^2}{5\rho\tau}}, \quad \text{if } 4\rho\tau \leq m \leq 2\rho\tau.$$

For skew-Hermitian matrices:

$$\epsilon_m \leq 12e^{-\left(\frac{\rho\tau}{m}\right)^2} e^{\rho\tau m}, \quad \text{if } m \geq 2\rho\tau.$$

For non-symmetric matrices with eigenvalues in the disk $|z + \rho| \leq \rho$, the error satisfies:

$$\epsilon_m \leq 12e^{-\rho\tau} e^{\rho\tau m}, \quad \text{if } m \geq 2\rho\tau.$$

3. Validation and Comparison. The algorithm was implemented in Python using NumPy and SciPy for linear algebra operations and sparse matrix support. SciPy's `scipy.sparse.linalg.expm_multiply` was used as a benchmark for validating both the accuracy and performance of the Krylov-based exponential approximation.

The implementation was compared against other common techniques for computing e^{tA}, including direct evaluation for small matrices via `scipy.linalg.expm`, Padé's approximation, as well as Taylor approximations. Evaluation focused on accuracy, assessed by computing the norm of the difference between the Krylov approximation and the benchmark result.

To ensure comprehensive testing, three types of matrices were used. The first was a small dense matrix, where A is a random 10×10 matrix and compared to `scipy.sparse.linalg.expm_multiply`. The second was a symmetric negative definite matrix. The third involved a sparse 2D Laplacian matrix. These test problems span a range of spectral and structural properties, providing a robust basis for evaluating the method's effectiveness.

4. Results. Presented are the results for the three different matrices tested: small dense matrix, symmetric negative definite matrix, and sparse 2D Laplacian matrix.

We first compared the relative errors of `scipy.linalg.expm` with `scipy.sparse.linalg.expm_multiply`. For the small dense matrix, the direct `scipy.linalg.expm` method compared with the `scipy.sparse.linalg.expm_multiply` method gave a relative error of 2.55×10^{-16}. For the the symmetric negative definite matrix, the direct `scipy.linalg.expm` method compared with the `scipy.sparse.linalg.expm_multiply` method gave a relative error of 7.08×10^{-16}. Lastly, for sparse 2D Laplacian matrix, the direct `scipy.linalg.expm` method compared with the `scipy.sparse.linalg.expm_multiply` method compared with gave a relative error of 1.64×10^{-14}.

Tables 1 and 2 contain the relative errors of the Krylov subspace method and Taylor series calculated using the norm of the small dense matrix at $t = 1.0$. Fig 1 displays the relative error as a function of the Krylov subspace dimension or number of Taylor terms for the Krylov subspace method and Taylor series.

TABLE 1
Results for Small Dense Matrix. *Relative errors (Frobenius norm) at* $t = 1.0$

Method	**Dimension/Terms**			
	1	**3**	**5**	**7**
Krylov (m)	$9.40 * 10^{-2}$	$1.48 * 10^{-4}$	$7.99 * 10^{-8}$	$2.05 * 10^{-11}$
Taylor	$9.41 * 10^{-2}$	$1.51 * 10^{-4}$	$8.59 * 10^{-8}$	$2.21 * 10^{-11}$

TABLE 2
Results for Small Dense Matrix. *Continued: Relative errors at* $t = 1.0$

Method	**Dimension/Terms**			
	9	**11**	**13**	**15**
Krylov (m)	$2.69 * 10^{-15}$	$2.58 * 10^{-16}$	$2.58 * 10^{-16}$	$2.58 * 10^{-16}$
Taylor	$3.49 * 10^{-15}$	$1.91 * 10^{-16}$	$1.91 * 10^{-16}$	$1.91 * 10^{-16}$

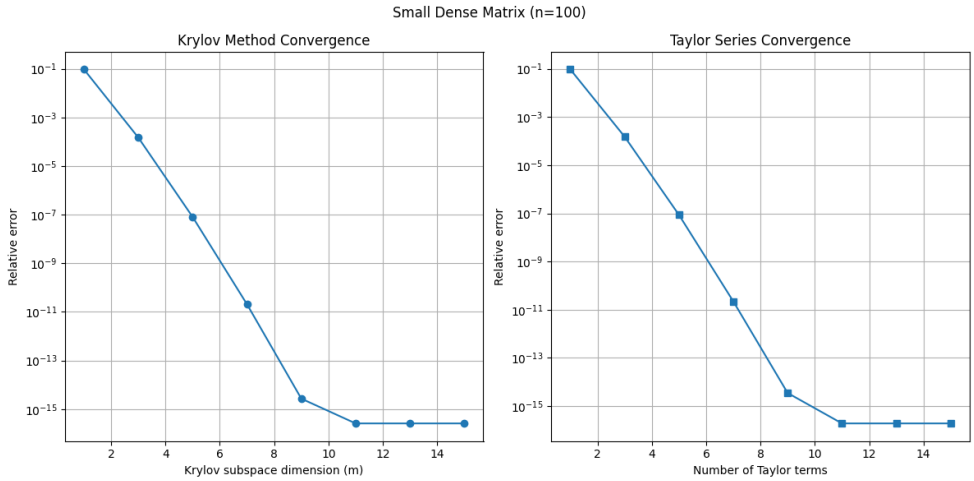


FIG. 1. *Convergence comparison at* $t = 1.0$ *for small dense matrix. (Generated by matplotlib)*

Tables 3 and 4 contain the relative errors of the Krylov subspace method and Taylor series calculated using the norm for the symmetric negative definite matrix at $t = 0.1$. Fig 2 displays the relative error as a function of the Krylov subspace dimension or number of Taylor terms for the Krylov subspace method and Taylor series.

TABLE 3
Results for Symmetric Negative Definite Matrix. *Relative errors (Frobenius norm) at* $t = 0.1$

Method	**Dimension/Terms**				
	10	**15**	**20**	**25**	**30**
Krylov (m)	$1.33 * 10^{-4}$	$2.98 * 10^{-8}$	$1.52 * 10^{-12}$	$1.68 * 10^{-15}$	$1.68 * 10^{-15}$
Taylor	$1.89 * 10^{3}$	$5.41 * 10^{2}$	$2.87 * 10^{1}$	$4.34 * 10^{-1}$	$2.42 * 10^{-3}$

TABLE 4
Results for Symmetric Negative Definite Matrix. *Continued: Relative errors at* $t = 0.1$ *(higher dimensions)*

Method	**Dimension/Terms**				
	35	**40**	**45**	**50**	**55**
Krylov (m)	$1.68 * 10^{-15}$	$1.68 * 10^{-15}$	$1.68 * 10^{-15}$	$1.68 * 10^{-15}$	$1.68 * 10^{-15}$
Taylor	$5.88 * 10^{-6}$	$7.03 * 10^{-9}$	$4.55 * 10^{-12}$	$1.61 * 10^{-13}$	$1.61 * 10^{-13}$

FIG. 2. *Convergence comparison at $t = 0.1$ for symmetric negative definite matrix. (Generated by matplotlib)*

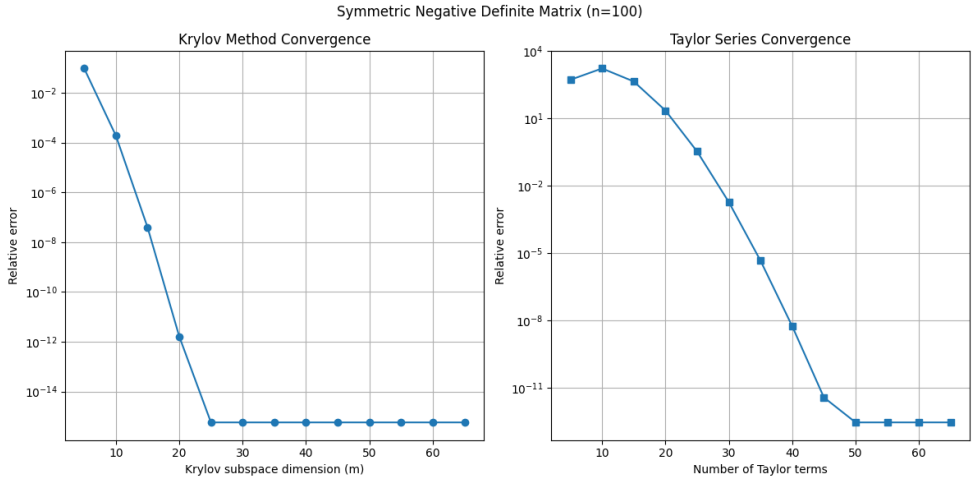


TABLE 5

Results for Sparse 2D Laplacian Matrix. *Relative errors (Frobenius norm) at $t = 0.01$*

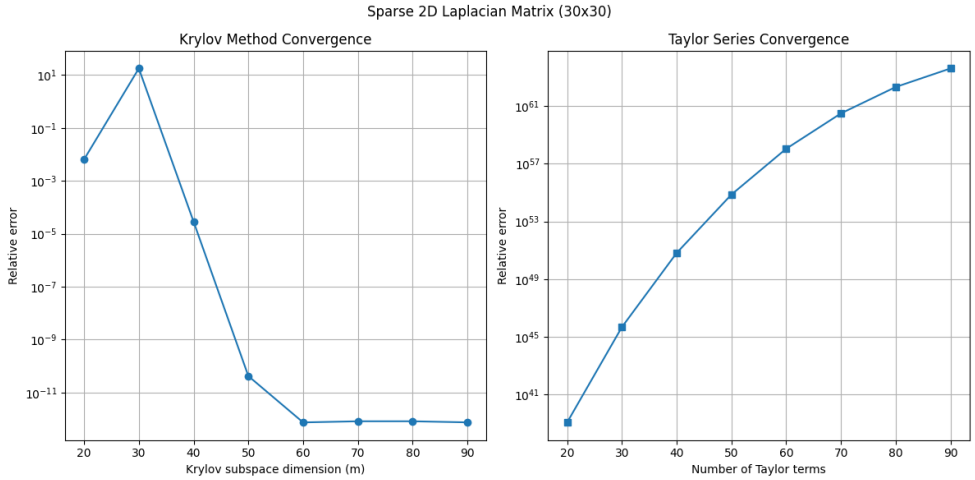
Method	**Dimension/Terms**			
	20	**30**	**40**	**50**
Krylov (m)	$6.42 * 10^{-3}$	$1.76 * 10^{1}$	$2.85 * 10^{-5}$	$4.18 * 10^{-11}$
Taylor	$1.16 * 10^{39}$	$4.71 * 10^{45}$	$6.53 * 10^{50}$	$7.30 * 10^{54}$

TABLE 6

Results for Sparse 2D Laplacian Matrix. *Continued: Relative errors at $t = 0.01$ (higher dimensions)*

Method	**Dimension/Terms**			
	60	**70**	**80**	**90**
Krylov (m)	$7.41 * 10^{-13}$	$8.18 * 10^{-13}$	$8.18 * 10^{-13}$	$7.44 * 10^{-13}$
Taylor	$1.09 * 10^{58}$	$3.09 * 10^{60}$	$2.09 * 10^{62}$	$4.04 * 10^{63}$

FIG. 3. *Convergence comparison at $t = 0.01$ for sparse 2D Laplacian matrix.(Generated by matplotlib)*



Tables 5 and 6 contain the relative errors of the Krylov subspace method and Taylor series calculated using the norm for the sparse 2D Laplacian matrix at $t = 0.01$. Fig 3 displays the relative error as a function of the Krylov subspace dimension or number of Taylor terms for the Krylov subspace method and Taylor series.

5. Discussion. For a randomly generated small dense matrix with $t = 1.0$, we observe that as the Krylov subspace dimension and the number of Taylor series terms increase, the relative error decreases and eventually converges to a minimum. Initially, the Krylov subspace method exhibits a lower relative error than the Taylor series. However, once both methods converge to a minimum, the Taylor series achieves slightly higher accuracy, reaching a final relative error of 1.91×10^{-16} compared to 2.58×10^{-16} for the Krylov method. The relative error of `scipy.linalg.expm`, compared to `scipy.sparse.linalg.expm_multiply`, is 2.55×10^{-16}—greater than that of the Taylor series and close to that of the Krylov method. This suggests that the Taylor series, given a sufficient number of terms, can slightly outperform both Krylov and `scipy.linalg.expm` in terms of accuracy for small dense matrices.

For a symmetric negative definite matrix with $t = 0.1$, both the Krylov and Taylor methods show decreasing relative errors as the Krylov dimension or number of terms increases. The Krylov method converges more quickly and ultimately achieves a lower relative error of 1.68×10^{-15}, compared to 1.61×10^{-13} for the Taylor series. Compared to the small dense case, both methods require larger Krylov dimensions or more Taylor terms to achieve convergence. The relative error of `scipy.linalg.expm` compared to `scipy.sparse.linalg.expm_multiply` is 7.08×10^{-16}, which is lower than the final errors of both Krylov and Taylor methods in this setting.

For the sparse 2D Laplacian matrix with $t = 0.01$, the Krylov method exhibits decreasing relative error and eventually converges as the subspace dimension increases. In contrast, the Taylor series fails to converge and instead diverges as more terms are added. This divergence is clearly illustrated in the corresponding error plot. The Krylov method requires significantly higher subspace dimensions to achieve convergence in this setting. The relative error of `scipy.linalg.expm` compared to `scipy.sparse.linalg.expm_multiply` is 1.64×10^{-14}—lower than both the Krylov and Taylor methods. These results highlight the instability of Taylor series for this type of matrix and the slower convergence of the Krylov method in this situation.

6. Conclusion. This technical report highlights the strengths and limitations of different approaches for computing matrix exponentials across various matrix types. Based on our findings, we offer the following recommendations:

- Small Dense Matrices: Both the Krylov subspace method and the Taylor series perform well, with the Taylor series achieving slightly higher accuracy at high orders. However, the Krylov method reaches acceptable accuracy more quickly and with less computational cost. Thus, we recommend using Krylov for efficiency and Taylor series only when maximal precision is required and performance is less critical.
- Symmetric Negative Definite Matrices: The Krylov method outperforms the Taylor series in both convergence rate and final error. It is the preferred approach in this case. For even higher precision, `scipy.linalg.expm` is a strong alternative, offering the lowest relative error among the tested methods.
- Sparse Matrices (e.g., 2D Laplacian): The Taylor series diverges, and the Krylov method converges slowly. In this regime, `scipy.sparse.linalg.expm_multiply` is recommended due to its stability, efficiency, and ability to handle large sparse matrices without forming the full exponential.

Overall, the Krylov subspace method seems to be a versatile and robust choice, especially for large or structured problems. While direct methods like `scipy.linalg.expm` are preferred for small matrices requiring high precision, and functions like `expm_multiply` excel for large sparse systems, the Krylov subspace method seems a balance between accuracy, efficiency, and scalability.

Future directions may include adaptive Krylov dimension selection, preconditioning

techniques, or hybrid methods (e.g., rational Krylov) to further enhance performance and robustness in challenging settings. Specifically, adaptive approaches could dynamically adjust subspace sizes using residual-based error estimators, while spectral preconditioning (e.g., shift-and-scale transformations) may accelerate convergence for stiff systems. Finally, GPU-optimized implementations of these techniques could leverage parallel architectures for large-scale problems, which helps to bridge the gap between theoretical convergence rates and practical performance.

REFERENCES

[1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel, *Implementation of a restarted Krylov subspace method for the evaluation of matrix functions*, Linear Algebra Appl., 429 (2008), pp. 2293–2314, https://doi.org/10.1016/j.laa.2008.06.029.

[2] S. Güttel, *Rational Krylov Methods for Operator Functions*, PhD thesis, Technische Universität Berlin, 2013.

[3] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925, https://doi.org/10.1137/S0036142995280572.

[4] M. L. Liou, *A novel method of evaluating transient response*, Proc. IEEE, 54 (1996), pp. 20–23.

[5] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49, https://doi.org/10.1137/S00361445024180.

[6] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228, https://doi.org/10.1137/0729014.

[7] J. R. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, technical report, Carnegie Mellon University, 1994, https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

[8] L. N. Trefethen, *Pseudospectra of linear operators*, Numerical Analysis Report NA-96/65, Oxford University Computing Laboratory, 1996, https://people.maths.ox.ac.uk/trefethen/publication/PDF/1996_65.pdf.