RWTH Aachen University

Institute for Geometry and Practical Mathematics (IGPM)

Randomized Low-Rank Runge-Kutta Methods

Bachelor's Thesis

Submitted by

Taha Atlagh
taha.atlagh@rwth-aachen.de

Supervised by

Prof. Dr. rer. nat. Bachmayr
Prof. Dr. rer. nat. Oster

September 30, 2025

Contents

1. Introduction

Large-scale matrix differential equations are ubiquitous in science and engineering, arising from the discretization of partial differential equations or from models in fields such as quantum mechanics, uncertainty quantification, and machine learning [**15**; **18**; **21**]. We consider autonomous systems of the form

$$\dot{A}(t) = F(A(t)), \quad A(0) = A_0 \in \mathbb{R}^{m\times n}, \tag{1.1}$$

where the dimensions m and n are large. For such systems, standard numerical integration schemes become computationally prohibitive due to the immense cost of storing and manipulating the full solution matrix $A(t)$ at each time step.

A prominent strategy to overcome this is **dynamical low-rank approximation**, which constrains the solution to evolve on the manifold $\mathcal{M}_r$ of rank-r matrices, where $r \ll \min(m, n)$. According to the Dirac-Frenkel variational principle [**31**; **16**], the optimal low-rank approximation $Y(t) \in \mathcal{M}_r$ is governed by the projection of the vector field onto the tangent space of the manifold

$$\dot{Y}(t) = P_{T_{Y(t)}\mathcal{M}_r}(F(Y(t))), \quad Y(0) = Y_0 \in \mathcal{M}_r. \tag{1.2}$$

Specialized integrators, such as **Projected Runge-Kutta (PRK) methods** [**3**], have been developed to solve this system efficiently. However, their convergence guarantees rely on a strong **tangentiality condition**, which assumes that the vector field $F(Y)$ lies almost entirely within the tangent space $T_{Y(t)}\mathcal{M}_r$. This condition can be violated in many practical scenarios—for instance, when the manifold exhibits high-frequency oscillations near the solution—leading to a severe loss of accuracy.

This thesis analyzes low-rank time integration methods that circumvent this restrictive assumption. Instead of a deterministic tangent space projection, our approach employs a **randomized projection** at each intermediate stage of a Runge-Kutta method. This prevents the uncontrolled rank growth that would otherwise impair efficiency. The rank reduction is achieved using the generalized Nyström method [**19**], which is both numerically robust and computationally efficient. Building upon the foundational framework of randomized low-rank algorithms [**5**], we investigate sketching with both dense Gaussian matrices and structured random matrices, particularly the Subsampled Randomized Fourier Transform (SRFT), to further accelerate the computation. The primary goal is their implementation, analysis, and validation. Our contributions include

- A probabilistic error analysis that links the convergence order to the so-called *stage order* of the underlying RK method.
- Numerical experiments which indicate that RLRK methods often achieve the full classical convergence order of the base RK method in practice, which can be higher than the stage order analysis suggests.
- An investigation into computational efficiency by implementing the sketching process with both dense Gaussian matrices and the computationally cheaper **Subsampled Randomized Fourier Transform (SRFT)**, demonstrating its superior scalability.
- Beyond fixed-step integrators, this work implements and evaluates an **adaptive RLRK integrator** based on the Dormand-Prince 5(4) method (**Rand Dopri45**). Its robustness and efficiency are compared against its fixed-step counterparts using work-precision diagrams.

Our theoretical analysis is founded on three assumptions. The first two are standard in the analysis of low-rank methods [**3**; **12**; **13**], while the third adapts the low-rank approximability condition from the seminal work on RLRK methods [**1**].

Assumption 1 (Lipschitz continuity of F)**.** *The function F is assumed to be Lipschitz continuous, meaning there exists a constant $L > 0$ such that*

$$\|F(X) - F(Y)\|_F \le L\|X - Y\|_F \quad \textit{for all } X, Y \in \mathbb{R}^{m\times n}. \tag{1.3}$$

The Picard–Lindelöf theorem then guarantees the local existence and uniqueness of the solution.

Assumption 2 (Smoothness of the flow map)**.** *Let Φ_F^t be the exact flow of the differential equation, such that $A(t) = \Phi_F^t(A_0)$. For a numerical method of order τ, we assume the derivatives up to order $\tau + 1$ are uniformly bounded:*

$$\left\| \frac{d^{j+1}}{dt^{j+1}} \Phi_F^t(Y) \right\|_F \le C_j, \quad j = 0, 1, \ldots, \tau, \tag{1.4}$$

for all $Y \in \mathbb{R}^{m\times n}$. This condition is essential for the local error analysis of higher-order methods.

Assumption 3 (Low-rank approximability of the flow)**.** *The solution is assumed to remain close to the low-rank manifold over short time intervals. Formally, for every $h \leq h_0$, we assume*

$$\left\|\Phi_F^h(Y) - [[\Phi_F^h(Y)]]_r\right\|_F \leq C_M h\epsilon \quad \text{for all } Y \in \mathcal{M}_r, \tag{1.5}$$

where $[[\cdot]]_r$ denotes the best rank-r approximation and C_M is a constant. This condition on the flow is significantly weaker than the tangentiality condition on the vector field required by PRK methods.

For the sake of a clear theoretical presentation, we assume that Assumptions 1.3, 1.4, and 1.5 hold globally, which significantly simplifies the analysis. While it is common in the literature to impose such conditions only locally within a neighborhood of the exact solution $A(t)$, this work adopts a global framework. The challenge with randomized methods is the small but non-zero probability that the approximation may leave any such neighborhood. We later discuss in Remark 4.2 how our analysis can be adapted to this local setting, which results in slightly weaker error estimates.

The remainder of this thesis is organized as follows. Chapter 2 establishes the fundamentals of classical Runge-Kutta methods, including adaptive step-size control. Chapter 3 introduces the theory of randomized low-rank approximation, detailing the two-stage process of subspace construction and factorization via the Nyström method, and discusses the efficiency gains from using SRFT matrices. Chapter 4 synthesizes these concepts to present the RLRK algorithm, along with a detailed error analysis. Finally, Chapter 5 provides a range of numerical experiments to validate the theoretical results, comparing RLRK with projected RK methods, evaluating the performance of the adaptive Rand Dopri45 solver, and testing the methods on problems including the Lyapunov, non-linear Schrödinger, and Allen-Cahn equations.

Remark 1.1 (Computational Environment). All numerical experiments were conducted on a 2024 MacBook Air equipped with an Apple M4 chip. The implementation was carried out in Python version 3.12, utilizing the scientific computing libraries NumPy (version 1.26) for numerical operations and Matplotlib (version 3.8) for data visualization. The numerical experiments detailed in Chapter 5, however, were predominantly programmed and executed in MATLAB (version R2025a).

2. Runge-Kutta Methods

The presentation in this section largely follows [**6**; **7**]; rigorous proof of the main results can be found in [**8**]. We consider vector-valued initial value problems with solutions $y : [0, T) \to \mathbb{R}^n$, where $n \in \mathbb{N}$. The right-hand side function f is defined on domains of the form $[0, T) \times J$, where $T > 0$ and $J \subseteq \mathbb{R}^n$ is an open set. For simplicity's sake, unless otherwise stated, we will use the starting value $t_0 = 0$.

Definition 2.1 (Initial value problem)**.** Let $J \subseteq \mathbb{R}^n$ be an open set and $y_0 \in J$. A function $y : [0, T) \to J$ is called a solution of the initial value problem

$$y'(t) = f(t, y(t)), \quad y(0) = y_0, \tag{2.1}$$

on $[0, T)$ if it is differentiable, satisfies the differential equation for all $t \in [0, T)$, and meets the initial condition. We also use the shorthand notation $y' = f(t, y)$ on $[0, T)$.

The initial value problem can be directly converted into an integral equation, providing an alternative but equivalent representation.

Proposition 2.2. *Consider a continuous function $f : [0, T) \times J \to \mathbb{R}$, where $J \subseteq \mathbb{R}^n$ is an open set containing a point y_0. A function $y : [0, T) \to J$ is a continuously differentiable solution to the initial value problem*

$$y'(t) = f(t, y(t)), \quad y(0) = y_0$$

if and only if y is a continuous solution to the integral equation

$$y(t) = y_0 + \int_0^t f(s, y(s))\, ds. \tag{2.2}$$

Proof. ($\Rightarrow$) Assume $y \in \mathcal{C}^1([0, T), J)$ is a solution to the initial value problem. Integrating the differential equation from 0 to t gives

$$y(t) - y(0) = \int_0^t f(s, y(s))\, ds.$$

Using the initial condition $y(0) = y_0$, we obtain the integral equation

$$y(t) = y_0 + \int_0^t f(s, y(s))\, ds.$$

Since $y \in \mathcal{C}^1([0, T), J)$, it is also continuous.

($\Leftarrow$) Assume $y \in \mathcal{C}([0,T), J)$ is a solution to the integral equation. First, evaluating at $t = 0$ yields

$$y(0) = y_0 + \int_0^0 f(s, y(s))\, ds = y_0.$$

The initial condition is satisfied. Second, since f and y are continuous, their composition $s \mapsto f(s, y(s))$ is also continuous. By the Fundamental Theorem of Calculus, we can differentiate the integral equation

$$y'(t) = f(t, y(t)).$$

Since $y'(t)$ is equal to the continuous function $f(t, y(t))$, y' itself must be continuous. Therefore, y is a continuously differentiable solution to the IVP. □

While this integral equation correctly represents the IVP, it is an implicit formulation, not an explicit solution, since the unknown function y appears within the integrand. The fundamental question of whether a solution exists and is unique is answered by the following theorem, which guarantees it, provided f is continuous and locally Lipschitz continuous in its second argument.

Theorem 2.3 (Picard–Lindelöf)**.** *Let f be continuous on $D = [0,T) \times J$, where $J \subset \mathbb{R}^n$ is open. For every $(t_0, z_0) \in D$, suppose there exists an open neighborhood U and a constant $L > 0$ such that*

$$\|f(t,y) - f(t,z)\| \leq L\|y - z\|, \quad \text{for all } (t,y),(t,z) \in U \cap D.$$

Then, for every initial value $y_0 \in J$, there exists a time $T_0 > 0$ and a unique continuously differentiable solution $y : [0, T_0) \to J$ to the IVP. Furthermore, the solution curve $(t, y(t))$ admits a unique extension up to the boundary of D.

This result is sufficient for our purposes, as we will always assume (or require) that f is Lipschitz continuous in later sections.

While the Picard-Lindelöf theorem guarantees the existence and uniqueness of solutions, it does not provide a direct method for finding them numerically. For this purpose, we turn to numerical schemes, among which Runge-Kutta methods are widely employed. Their construction is based on applying numerical quadrature to the integral formulation given in 2.2.

Applying this procedure iteratively to the subsequent subintervals generates the complete numerical solution. We first discretize the time domain into a series of points $t_0, t_1, t_2, \ldots$, defined by

$$t_i = t_0 + ih \quad \text{for } i = 0, 1, 2, \ldots$$

where t_0 is the initial time and h is a fixed step size. The method's evolution rule advances the solution from the current time point t_i to the next, t_{i+1}, by approximating the integral using an s-stage rule

$$\int_{t_i}^{t_{i+1}} f(t, y(t))\, dt \approx h \sum_{j=1}^{s} b_j f(t_i + c_j h, \eta_j),$$

with suitable coefficients b_j, $c_j \in \mathbb{R}$ and intermediate values $\eta_j \approx y(t_i + c_j h)$.

However, the intermediate values η_j are not yet defined in a computable way. The principle of Runge-Kutta methods is to approximate these stages themselves using a similar weighted sum of function evaluations. Combining these two aspects leads to the general approach for Runge-Kutta methods.

Definition 2.4 (Runge-Kutta Method)**.** An s-stage Runge-Kutta method approximates the solution of the initial value problem (2.1) by the following scheme

$$\eta_j = y_i + h \sum_{k=1}^{s} a_{jk} f(t_i + c_k h, \eta_k), \qquad \sum_{k=1}^{s} a_{jk} = c_j, \quad j = 1, \ldots, s, \tag{2.3}$$

$$y_{i+1} = y_i + h \sum_{j=1}^{s} b_j f(t_i + c_j h, \eta_j), \qquad \sum_{j=1}^{s} b_j = 1. \tag{2.4}$$

The values η_j are called the internal stages. If the coefficients satisfy $a_{jk} = 0$ for all $k \geq j$, the method is called *explicit*, otherwise it is *implicit*.

The structure of the matrix A determines the computational approach for a Runge-Kutta method. Explicit methods allow for an efficient, sequential evaluation of the stages due to A being strictly lower triangular. Implicit methods, however, require solving a coupled system of equations at each step. This increased effort is justified by their superior stability, which is important for stiff problems (i.e., problems where stability, rather than accuracy, forces explicit methods to take impractically small step sizes).

Due to their favorable computational cost, this thesis will focus exclusively on explicit Runge-Kutta methods.

The coefficients of an RK method are conventionally represented in a compact form known as the Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \quad \text{where} \quad c = (c_1, \ldots, c_s)^T, \quad A = (a_{jl})_{j,l=1}^{s}, \quad b^T = (b_1, \ldots, b_s).$$

TABLE 1. The Butcher Tableau for a Runge-Kutta method.

The following tableaus represent some foundational RK methods.

Example 2.5. Explicit Euler Method (Order 1):

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

Implicit Midpoint Rule (Order 2):

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

Heun's Method (Explicit Trapezoidal Rule, Order 2):

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}$$

The Classical Runge-Kutta Method (RK4, Order 4):

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

The examples presented, from the first-order Euler method to the fourth-order classical Runge-Kutta method, indicate that not all methods have the same accuracy. This naturally leads to the need for a formal measure of a method's quality, which is provided by the following definition and theorem.

Definition 2.6 (Consistency Order)**.** A one-step method has a (consistency) order of $q \in \mathbb{N}$ if for every solution $y : I \to J$ of the initial value problem (IVP) with $y_0 \in J$ and $f \in C^q(I \times J)$, there exist a constant $c > 0$ and an $h_0 > 0$ such that for all $t_i \in I$ and $y_i = y(t_i)$, the following holds:

$$\|y_{i+1} - y(t_i + h)\|_2 \leq ch^{q+1}, \quad \text{for } 0 \leq h \leq h_0. \tag{2.5}$$

The consistency order of the method is linked to the accuracy of its underlying quadrature rule.

Theorem 2.7. *If a one-step method of the form*

$$y_{i+1} = y_i + h \sum_{j=1}^{s} b_j f(t_i + c_j h, \eta_j) \tag{2.6}$$

has a consistency order of q, then the corresponding quadrature formula

$$Q[g] = \sum_{j=1}^{s} b_j g(c_j) \approx \int_0^1 g(x)\, dx \tag{2.7}$$

has a degree of exactness of at least $q - 1$.

Proof. The proof can be found in [**35**], Theorem 76.4. □

While higher-order methods provide greater accuracy for a given step size, achieving overall efficiency often involves more than simply choosing a method with a high consistency order. In practice, the smoothness of the solution can vary over the integration interval. A fixed step size small enough to ensure accuracy in regions of rapid change may be unnecessarily costly in areas where the solution is smooth. This practical consideration motivates a shift from fixed-step integration to methods that can dynamically adjust the step size to maintain a desired level of accuracy.

2.1. **Adaptive Step Size Control.** In many practical applications, adaptive step size control is sufficient to achieve a desired accuracy without resorting to implicit methods. Since the exact solution is generally unknown, one compares two numerical approximations of different orders to estimate the local error. This leads to the construction of embedded Runge–Kutta methods, which provide an efficient way to control the error without additional evaluations of the right-hand side function f.

An embedded method consists of a primary Runge–Kutta method of order q, and a secondary (control) method of order $\hat{q}$, both sharing the same internal stages η_j and coefficients $A = (a_{ij})$. Given the internal stages $\eta_j \approx y(t_i + c_j h)$, the two approximations are computed as

$$y_{i+1} = y_i + h \sum_{j=1}^{s} b_j f(t_i + c_j h, \eta_j), \tag{2.8}$$

$$\hat{y}_{i+1} = y_i + h \sum_{j=1}^{s} \hat{b}_j f(t_i + c_j h, \eta_j). \tag{2.9}$$

The structure of an embedded Runge–Kutta method with a primary method of order q and a control method of order $\hat{q}$ is typically represented by the following extended Butcher tableau:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \\ & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_s \end{array}$$

The difference between the two approximations serves as an estimate of the local truncation error, if $\hat{q} = q + 1$

$$\begin{aligned} \|y_{i+1} - \hat{y}_{i+1}\| &= (y_{i+1} - y(t_{i+1})) + (y(t_{i+1}) - \hat{y}_{i+1}) \\ &= Ch^{q+1} + \mathcal{O}(h^{q+2}). \end{aligned}$$

Assuming a sufficiently small step size h, one obtains the following estimate

$$\delta_{i+1}(h) := \|y_{i+1} - \hat{y}_{i+1}\| \approx Ch^{q+1}. \tag{2.10}$$

The step is accepted if $\delta_{i+1} \leq \varepsilon$, where $\varepsilon > 0$ is a user-specified tolerance. Otherwise, the step is rejected, and the computation is repeated with a smaller step size. Assuming the error behaves as $\delta_{i+1}(h) \approx Ch^{q+1}$, one can estimate a new step size $\tilde{h}$ such that the expected error in the next step satisfies $\delta_{i+1}(\tilde{h}) \leq \varepsilon$. We obtain

$$\delta_{i+1}(\tilde{h}) \approx C\tilde{h}^{q+1} = \left(\frac{\tilde{h}}{h}\right)^{q+1} Ch^{q+1} \approx \left(\frac{\tilde{h}}{h}\right)^{q+1} \delta_{i+1}(h). \tag{2.11}$$

This leads to the condition

$$\left(\frac{\tilde{h}}{h}\right)^{q+1} \delta_{i+1}(h) \leq \varepsilon. \tag{2.12}$$

Solving this inequality for h provides an upper bound for the new step size

$$\tilde{h} \leq \left(\frac{\varepsilon}{\delta_{i+1}(h)}\right)^{\frac{1}{q+1}} h. \tag{2.13}$$

To ensure robustness and prevent frequent step rejections, the new step size is chosen conservatively

$$\tilde{h} = \tau \left(\frac{\varepsilon}{\delta_{i+1}(h)}\right)^{\frac{1}{q+1}} h \tag{2.14}$$

where $\tau \in (0, 1]$ is a safety factor, later chosen as $\tau = 0.9$.

A notable optimization used in practice is the so-called FSAL trick (First Same As Last): the last stage k_s from the previous time step is reused as the first stage k_1 in the next step, thereby reducing the number of evaluations per step from s to $s - 1$.

The Dormand–Prince 5(4) method, first presented in [**5**], is one of the most widely used embedded Runge–Kutta methods and has become a standard choice for solving non-stiff initial value problems. It combines a fifth-order accurate Runge–Kutta method with a fourth-order control estimate, enabling adaptive step size control. For the Dormand–Prince method, the Butcher tableau is as follow.

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40}
\end{array}
$$

To compare the theoretical properties and practical performance of Runge-Kutta methods with fixed and adaptive step-size control, a numerical experiment is conducted in this section. The objective is to evaluate the efficiency of the first-order explicit Euler method (RK1) and the classic fourth-order Runge-Kutta method (RK4) against the adaptive Dormand-Prince 5(4) method Efficiency is defined as the achieved accuracy in relation to the computational cost. The following numerical experiment on the efficiency of Runge-Kutta methods is based on an example from source [**29**].

2.2. **Test Problem: The Arenstorf Orbit.** To assess the practical performance of the numerical methods, the Arenstorf orbit, a well-known problem from celestial mechanics, is used. It describes the trajectory of a body of negligible mass in the gravitational field of two larger bodies, such as the Earth and the Moon, in a rotating coordinate system. This is a special solution to the restricted three-body problem. The motion is governed by the following system of second-order ordinary differential equations:

$$
\begin{aligned}
y_1'' &= y_1 + 2y_2' - \mu' \frac{y_1 + \mu}{D_1} - \mu \frac{y_1 - \mu'}{D_2} \\
y_2'' &= y_2 - 2y_1' - \mu' \frac{y_2}{D_1} - \mu \frac{y_2}{D_2}
\end{aligned}
$$

where $D_1 = ((y_1 + \mu)^2 + y_2^2)^{3/2}$ and $D_2 = ((y_1 - \mu')^2 + y_2^2)^{3/2}$. The mass parameter is chosen as $\mu = 0.012277471$, with $\mu' = 1 - \mu$. The problem is characterized by singularities at the locations of the two massive bodies, which poses a challenge for numerical solvers. The initial conditions are set as:

$$
\begin{aligned}
y_1(0) &= 0.994, \quad y_1'(0) = 0 \\
y_2(0) &= 0, \quad y_2'(0) = -2.0015851063790825
\end{aligned}
$$

These conditions result in a periodic orbit with period $T \approx 17.065$. The dynamic nature of the problem, with phases of slow and rapid change in the solution, makes it an ideal test case for comparing fixed-step and adaptive methods.

The experiment compares the performance of three solvers: the first-order forward Euler method (RK1), the classic fourth-order Runge-Kutta method (RK4), and the adaptive Dormand-Prince 5(4) method (DOPRI5). To quantify accuracy, a high-precision reference solution was generated using the implicit Runge-Kutta method *Radau* with very strict relative and absolute tolerances (10^{-13} and 10^{-14}, respectively). The use of an implicit method from a different solver class ensures a robust and independent benchmark.

The computational cost is measured by the total number of function evaluations of the right-hand side of the ODE system. The accuracy is determined by the Root Mean Square Error (RMSE) of the entire computed trajectory relative to a reference solution. This error is calculated at the discrete time steps generated by the respective solver during the simulation. The formula for the RMSE [**29**] is given by

$$
\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2}
$$

where N is the number of time steps, $\mathbf{y}_i$ is the computed solution vector, and $\hat{\mathbf{y}}_i$ is the reference solution vector at time t_i. For the work-precision analysis, the fixed-step methods (Euler and RK4) were executed with a range of 8 different step counts, logarithmically spaced from approximately 1,600 to 63,000 steps. The adaptive method (DOPRI5) was run with a spectrum of 8 decreasing error tolerances, where both the relative and absolute tolerances were varied logarithmically from 10^{-4} to 10^{-11}.

The results of the numerical comparison are presented in two figures. Figure 1 shows the computed trajectories, while Figure 2 presents the work-precision diagram, which plots accuracy against computational cost.

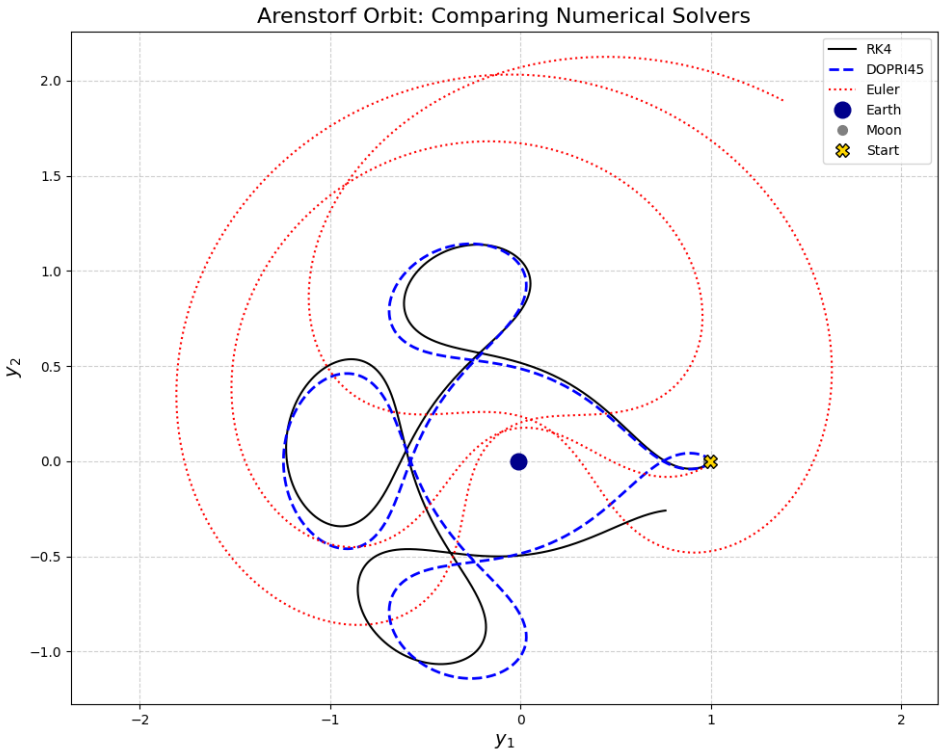

FIGURE 1. Comparison of the computed trajectories for the Arenstorf orbit.

2.2.1. *Trajectory Comparison.* The trajectories shown in Figure 1 illustrate a clear hierarchy of numerical accuracy. At the lowest end, the Euler method (RK1) fails clearly, despite 24,000 steps, its path diverges almost immediately, resulting in a qualitatively incorrect solution. The RK4 method (with 6,000 steps) performs better, correctly tracing the orbit for most of its duration. However, its accumulated error is still large enough to cause it to miss the starting point, failing to reproduce the closed orbit. Only the adaptive DOPRI5 method (with 796 steps) is shown to be sufficiently accurate, precisely capturing the complex, stable Arenstorf orbit from start to finish.

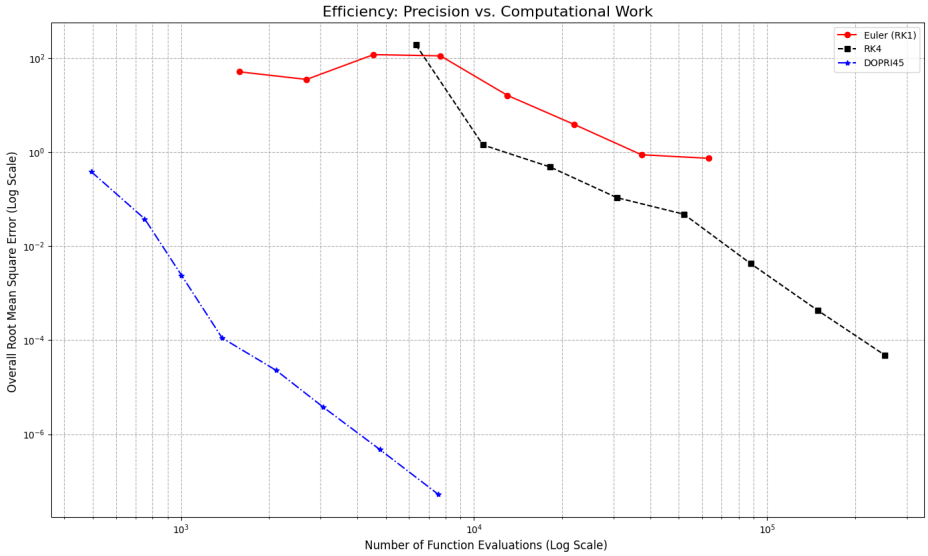

FIGURE 2. Work-precision diagram for the Euler, RK4, and DOPRI345 methods. The y-axis shows the RMSE, and the x-axis shows the number of function evaluations, both on a logarithmic scale.

2.2.2. *Efficiency Analysis (Work-Precision Diagram).* The work-precision diagram in Figure 2 provides a quantitative measure of solver efficiency. The most efficient methods are located in the bottom-left corner, as they achieve low error for low computational cost. The results lead to three observations:

a) **Euler Method (RK1)**: The curve for the Euler method is positioned in the upper-right region of the diagram, establishing it as the least efficient of the tested methods. It yields a high error even for a large number of function evaluations, making it unsuitable for this problem.
b) **Classic RK4 Method**: The RK4 method exhibits a significant increase in efficiency over the first-order method. Due to the logarithmic y-axis, the vertical distance between the curves represents a multiplicative factor in accuracy. Consequently, for a given computational cost, the RK4 method achieves an error that is several orders of magnitude lower than that of the Euler method. This result clearly illustrates the advantage of employing a higher-order fixed-step solver.
c) **Dormand-Prince 5(4) Method (DOPRI5)**: The adaptive DOPRI5 method demonstrates the highest efficiency, with its performance curve situated in the bottom-left of the diagram. For any target accuracy, it requires the fewest function evaluations, surpassing the performance of both fixed-step methods. This superior efficiency stems from its adaptive step-size control, which allocates computational effort more effectively by reducing the step size only in dynamically complex regions of the trajectory, such as when passing close to the Earth or the Moon.

In conclusion, the numerical experiment confirms that for problems with varying solution dynamics like the Arenstorf orbit, an adaptive Runge-Kutta method such as Dormand-Prince is vastly more efficient than its fixed-step counterparts.

2.3. **Matrix differential equations and time discretization.** In many scientific and engineering applications, the unknown variable in an initial value problem is not a vector but rather a matrix. We consider matrix differential equations of the form

$$A'(t) = F(t, A(t)), \quad A(t_0) = A_0,$$

where $A : [t_0, T) \to \mathbb{R}^{m\times n}$ is a time-dependent matrix, $A'(t)$ denotes the entry-wise derivative with respect to t, and $F : [t_0, T) \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is a given matrix-valued function. Such equations arise naturally in various fields, including physics (especially quantum mechanics) [**20**; **21**], control theory [**15**], and machine learning [**18**].

Conceptually, the application of Runge-Kutta methods to matrix differential equations is a straightforward extension of their use for vector-valued problems. If we consider the matrix $A(t)$ as a long vector obtained by stacking its columns (or rows) into $\mathrm{vec}(A(t)) \in \mathbb{R}^{mn}$, then the matrix differential equation can be recast as a large-scale vector ODE

$$\mathrm{vec}(A'(t)) = \mathrm{vec}(F(t, A(t))),$$

where $\mathrm{vec}(F(t, A(t)))$ acts as the right-hand side function for the vectorized system. Consequently, any standard Runge-Kutta method, such as those discussed in Section 2.1, can be applied directly to this vectorized system. For an s-stage explicit Runge-Kutta method applied to a matrix differential equation, the scheme takes the form:

$$K_1 = F(t_i, A_i) \tag{2.15}$$

$$K_2 = F(t_i + c_2 h, A_i + h a_{21} K_1) \tag{2.16}$$

$$\vdots \tag{2.17}$$

$$K_s = F(t_i + c_s h, A_i + h \sum_{j=1}^{s-1} a_{sj} K_j) \tag{2.18}$$

$$A_{i+1} = A_i + h \sum_{j=1}^{s} b_j K_j. \tag{2.19}$$

Here, A_i is the numerical approximation of $A(t_i)$, and K_j are the matrix-valued stages. The coefficients a_{ij}, b_j, and c_j are identical to those in the scalar/vector case and can be represented by the same Butcher tableau.

While conceptually simple, the direct application of standard Runge-Kutta methods to matrix differential equations faces significant computational challenges in high dimensions. Each stage evaluation of

$F(t, A)$ involves operations on large $m \times n$ matrices, and storing the intermediate results requires $\mathcal{O}(mn)$ memory. For large-scale systems, these demands quickly become computationally prohibitive.

To circumvent this limitation, we introduce a randomized low-rank approximation. This technique provides an efficient way to capture the essential information of a large matrix in a much smaller, low-rank format, thereby enabling the effective numerical integration of high-dimensional systems.

3. Randomized Low-Rank Approximation

While the optimal low-rank approximation is given by the Singular Value Decomposition (SVD), its computational cost is often prohibitive for large-scale problems. This motivates the use of randomized algorithms, which aim to find a near-optimal approximation at a fraction of the computational cost. The approach presented is conceptually divided into two stages. The first part of this chapter focuses on **Stage A**: the construction of a low-dimensional subspace that captures the essential action of a matrix, for which we follow the fundamental framework of Halko, Martinsson, and Tropp [**5**]. The second half of the chapter then addresses **Stage B**, where this subspace is used to compute a complete low-rank factorization. The subsequent integration of this two-stage process into the Runge-Kutta framework to form the RLRK method is based on the work of Lam, Ceruti, and Kressner [**1**] and will be the subject of Chapter 4.

To explain the algorithms, this chapter will initially concentrate on the use of dense Gaussian random matrices. Towards the end of the chapter, the *Subsampled Randomized Fourier Transform (SRFT)* will be introduced as an alternative in the form of a structured random matrix. The practical advantages and efficiency of the SRFT approach will then be investigated in the numerical experiments in Chapter 5.

3.1. **Intuition.**

Definition 3.1 (Standard Gaussian Matrix)**.** A random matrix $\Omega = (\omega)_{ij} \in \mathbb{R}^{m\times n}$ is a **standard Gaussian matrix** if its entries ω_{ij} are independent and identically distributed (i.i.d.) random variables following the standard normal distribution

$$\omega_{ij} \sim \mathcal{N}(0,1) \quad \text{for } i = 1,\dots,m \text{ and } j = 1,\dots,n. \tag{3.1}$$

To understand how randomness assists in solving the fixed-rank problem, we begin by considering a simple example given in [**5**]. Suppose we aim to find a basis for the range of a matrix A with exact rank k. To do this, we draw a random vector $\mathbf{v}$, and compute the product

$$y = A\mathbf{v}. \tag{3.2}$$

For the time being, the exact distribution of the random vector is not important. We simply treat y as a random sample from the range of A.

Next, we repeat this sampling process k times to generate the vectors

$$y^{(i)} = A\mathbf{v}^{(i)}, \quad i = 1, 2, \dots, k. \tag{3.3}$$

Due to the randomness, the set $\{\mathbf{v}^{(i)} : i = 1, 2, \dots, k\}$ of random vectors is likely to linearly independent and no linear combination will fall into the null space of A. As a result, the set

$$\{y^{(i)} : i = 1, 2, \dots, k\} \tag{3.4}$$

of sample vectors is also linearly independent, and thus spans the range of A. To obtain an orthonormal basis for the range, we simply need to orthonormalize the sample vectors. This technique effectively provides a method to find a basis for the range of A using random vectors, leveraging the property that random vectors, with high probability, will span the range of a matrix.

3.2. **Stage A: Constructing a Subspace via Randomized Sketch.**

The first phase of the randomized approximation process involves finding a low-dimensional subspace that accurately approximates the range of a given matrix A. This subspace is represented by an $m \times \ell$ orthonormal matrix Q, whose columns form a basis for the range. The algorithm used for this purpose, the so-called Randomized Range Finder, is presented in Algorithm 4.1.

The computational cost of the algorithm is determined by its three steps. The number of required floating-point operations can be estimated by the following relationship

$$T_{\text{basic}} \sim \ell n\, T_{\text{rand}} + \ell\, T_{\text{mult}} + \ell^2 m. \tag{3.5}$$

Algorithm 1: Randomized Range Finder

Input: An $m \times n$ matrix A and an integer ℓ. **Output:** An $m \times \ell$ orthonormal matrix Q.

(1) Generate an $n \times \ell$ Gaussian random matrix Ω.
(2) Compute the $m \times \ell$ matrix $Y = A\Omega$.
(3) Construct an $m \times \ell$ matrix Q whose columns form an orthonormal basis for the range of Y, for example, using a QR factorization $Y = QR$.

FIGURE 3. The Randomized Range Finder algorithm for finding an orthonormal basis for the range of a matrix using randomized projection.

Here, $\ell n T_{\text{rand}}$ represents the cost of generating the random matrix Ω, ℓT_{mult} accounts for the cost of the matrix-matrix multiplication $A\Omega$, and $\ell^2 m$ covers the orthogonalization in Step 3.

3.2.1. *Oversampling.* we now consider the case where

$$A = B + E, \tag{3.6}$$

where B is a rank-k matrix containing the information we are interested in, and E is a small perturbation. Our goal is to obtain a basis that spans the range of B as accurately as possible, rather than minimizing the number of basis vectors. In this case, we fix a small number p, and generate $k + p$ samples

$$y^{(i)} = A\mathbf{v}^{(i)} = B\mathbf{v}^{(i)} + E\mathbf{v}^{(i)}, \quad i = 1, 2, \ldots, k + p. \tag{3.7}$$

The perturbation E shifts each sample vector outside the range of B, which may prevent the set

$$\{y^{(i)} : i = 1, 2, \ldots, k\} \tag{3.8}$$

from covering the entire range of B. However, the enriched set

$$\{y^{(i)} : i = 1, 2, \ldots, k + p\} \tag{3.9}$$

has a much higher chance of spanning the required subspace, as the additional samples help to account for the perturbation E.

A potential issue with this approach is that the resulting matrix Y can become numerically ill-conditioned, meaning its columns are nearly linearly dependent. However, the algorithm is robust to this problem. The orthogonalization in Step 3 can reliably determine the numerical rank of matrix Y and automatically ignore the excess samples. This ensures that the resulting orthonormal basis Q is stable and consists only of the necessary linearly independent columns. Furthermore, the second phase of the approximation works even if the basis Q has a slightly larger dimension than the absolute minimum.

The key question is: how many extra samples are needed? Remarkably, for certain random sampling schemes, the failure probability decreases superexponentially with the oversampling parameter p. In practical terms, setting $p = 5$ or $p = 10$ often yields excellent results [**1**].

To further illustrate this concept, we will numerically demonstrate the impact of oversampling on the reliability of random sampling. By applying random sampling techniques to a rank-k matrix B perturbed by a small error matrix E, we will observe the difference in the quality of the basis obtained with and without oversampling.

Later in Section 5.2 we will conduct experiments with various values of p, the number of extra samples, and show how the probability of successfully spanning the subspace improves as p increases.

3.3. **Preliminaries, Low rank approximability and the tangent space projection.**

Transition to Dynamics. Before we proceed to **Stage B**, which details the construction of an explicit low-rank factorization, it is necessary to introduce foundational concepts for extending these ideas from static matrices to the setting of dynamical systems. The randomized methods of Stage A provide a subspace (a snapshot) capturing the essential information of a matrix at a single point in time. However, when dealing with matrix differential equations, the solution matrix $A(t)$ evolves continuously. A key challenge is therefore to understand how a low-rank approximation $Y(t)$ should evolve to stay close to $A(t)$.

This leads us to the geometry of low-rank matrices. The set of all $m \times n$ matrices of a fixed rank k forms **manifold**, which we denote by $\mathcal{M}_k$. For our approximation $Y(t)$ to remain on this manifold (i.e., to stay rank-k), its velocity vector, $\dot{Y}(t)$, must lie in the **tangent space** of the manifold at the point $Y(t)$, denoted by $T_{Y(t)}\mathcal{M}_k$.

Definition 3.2 (Tangent Space of the Manifold of Rank-k Matrices)**.** Let $\mathcal{M}_k$ be the manifold of $m \times n$ matrices of rank k. For a point $Y \in \mathcal{M}_k$ with a singular value decomposition $Y = U\Sigma V^T$ (where $U \in \mathbb{R}^{m\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$, $V \in \mathbb{R}^{n\times k}$), the **tangent space** $T_Y\mathcal{M}_k$ is the vector space describing the first-order variations around Y that remain within the manifold. It can be explicitly characterized as the set of all matrices $\delta Y \in \mathbb{R}^{m\times n}$ of the form:

$$\delta Y = dU \cdot \Sigma V^T + U \cdot d\Sigma \cdot V^T + U\Sigma \cdot dV^T$$

where $d\Sigma \in \mathbb{R}^{k\times k}$ is an arbitrary matrix, and $dU \in \mathbb{R}^{m\times k}$ and $dV \in \mathbb{R}^{n\times k}$ are matrices constrained by the condition that $U^T dU$ and $V^T dV$ must be skew-symmetric to preserve the orthonormality of the bases U and V.

The true dynamics, however, are given by the vector field $F(A(t))$, which does not necessarily respect this geometric constraint. A natural approach, known as the **Dirac-Frenkel variational principle**, is to project the true dynamics orthogonally onto the tangent space. The evolution of the low-rank approximation is then governed by the differential equation

$$\dot{Y}(t) = P_{Y(t)}(F(Y(t))), \quad \text{with } Y(0) = Y_0 \in \mathcal{M}_k \tag{3.10}$$

where $P_{Y(t)}$ is the orthogonal projector onto the tangent space $T_{Y(t)}\mathcal{M}_k$.

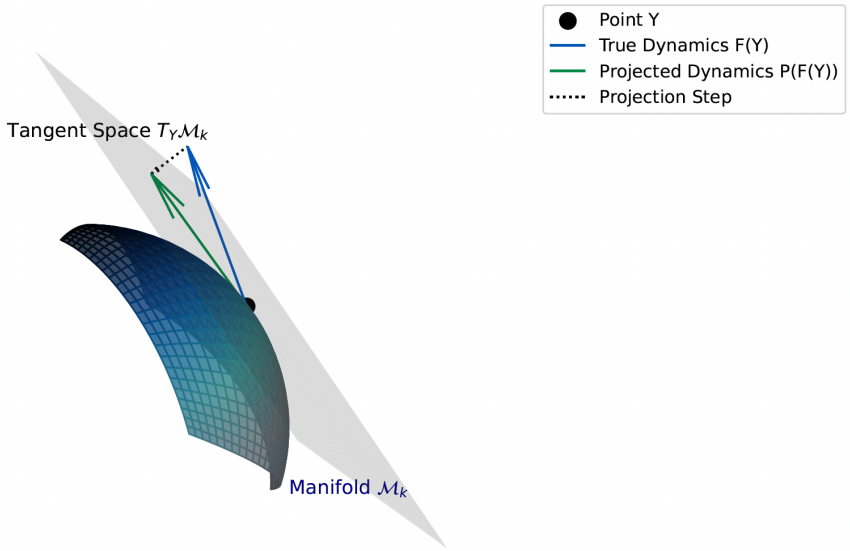

FIGURE 4. Visualization of the Dirac-Frenkel variational principle. The true dynamics vector $F(Y)$ (blue) is orthogonally projected onto the tangent space $T_Y M_k$ (gray plane) at point Y on the manifold M_k (blue surface). The result is the projected dynamics $P(F(Y))$ (green), which represents the best possible approximation of the true dynamics within the tangent space.

While this method yields a locally optimal approximation that ensures the trajectory remains on the manifold, the resulting solution $Y(t)$ is generally not the globally best approximation to $A(t)$. Its accuracy hinges on the **tangentiality condition**, how well the vector field $F(Y)$ is already contained within the tangent space. This idea is formalized by the following tangentiality condition, which requires the component of the vector field orthogonal to the tangent space to be small

$$\|F(Y) - P_r(Y)F(Y)\|_F \leq \tilde{\epsilon}, \quad \text{for all } Y \in \mathcal{M}_r \cap \{\text{suitable neighbourhood of } A(t)\}. \tag{3.11}$$

As noted in [**3**], this condition can be violated even when $A(t)$ is highly compressible, for instance, when the manifold $\mathcal{M}_k$ exhibits high-frequency oscillations near the solution or when the range and co-range of Y are contained in the orthogonal complement of the range and co-range of F(Y), respectively [**14**]. This phenomenon is demonstrated explicitly by the example detailed in 3.6.

Given the potential failure of the tangentiality condition, an alternative approach is to shift perspective from the properties of the vector field to the behavior of the flow itself.

3.3.1. *Relationship between Tangent Space Projection and Low-Rank Approximability.* In this section, we discuss our new approach the low rank approximability and the relation between Assumption 3.11 (the tangentiality condition of the vector field) and Assumption 1.5 (the low-rank approximability of the flow). We start by defining the flow.

Definition 3.3. Flow of a Matrix Differential Equation Let the space of $n \times n$ matrices be denoted by $M_n(\mathbb{R})$. Consider the initial value problem (IVP) for a matrix differential equation given by

$$\frac{dA(t)}{dt} = F(A(t)), \quad A(0) = A_0 \tag{3.12}$$

where $A(t) \in M_n(\mathbb{R})$ is the time-dependent matrix, $A_0 \in M_n(\mathbb{R})$ is the initial condition, and $F : M_n(\mathbb{R}) \to M_n(\mathbb{R})$ is a function that is sufficiently smooth (e.g., Lipschitz continuous) to guarantee the existence and uniqueness of a solution for some time interval.

The *flow* of the differential equation is a map Φ:

$$\Phi : D \to M_n(\mathbb{R}), \quad (t, A_0) \mapsto \Phi_t(A_0) \tag{3.13}$$

where $D \subseteq \mathbb{R} \times M_n(\mathbb{R})$ is the maximal domain on which the solution is defined. The map $\Phi_t(A_0)$ gives the unique solution $A(t)$ to the IVP at time t with initial condition A_0.

The flow must satisfy the following properties for all A_0 and all s, t for which it is defined:

(1) **Identity Property:** The flow at time $t = 0$ is the identity map.

$$\Phi_0(A_0) = A_0 \tag{3.14}$$

(2) **Group Property:** The evolution for a time $s + t$ is the composition of the evolution for time s and the evolution for time t.

$$\Phi_{s+t}(A_0) = \Phi_s(\Phi_t(A_0)) \tag{3.15}$$

This leads to the assumption of **low-rank approximability of the flow**. Instead of requiring the vector field F to be tangential, this condition only presumes that the exact solution of the differential equation, $A(t) = \Phi_t(A_0)$, does not stray far from the manifold $\mathcal{M}_k$ over short time intervals.

Formally, this is quantified by requiring the error of the best rank-k approximation to the exact flow to be small, i.e., assumption 1.5. This represents a weaker, but often more practical, condition as it allows the true dynamics to temporarily leave the manifold, a common occurrence in many applications. To formally establish that this assumption (low rank approcimability of the flow) is weaker, we introduce the following error estimate.

Lemma 3.4. *Given the assumptions 1.3 and 1.5, and with the error in the initial value bounded by $\|A_0 - Y_0\|_F \leq \delta$, the dynamical low-rank approximation yields an error bounded by*

$$\|Y(t) - A(t)\|_F \leq e^{Lt}\delta + \epsilon \int_0^t e^{Ls} ds. \tag{3.16}$$

To prove the error estimate for the dynamical low-rank approximation 3.4, we will make use of Gronwall's Lemma.

Theorem 3.5 (Gronwall's Lemma, Differential Form)**.** *Let $\varphi \in C^1(J, \mathbb{R})$ satisfy the inequality*

$$\varphi'(t) \leq \alpha(t) + \beta(t)\varphi(t) \quad \forall t \in J, \tag{3.17}$$

where $J = [t_0, t_1]$ and $\alpha, \beta \in C(J, \mathbb{R})$. Then for $t \in J$

$$\varphi(t) \leq \varphi(t_0) e^{\int_{t_0}^t \beta(\tau) d\tau} + \int_{t_0}^t \alpha(s) e^{\int_s^t \beta(\tau) d\tau} ds. \tag{3.18}$$

Now we can proof Lemma 3.4.

Proof. The derivative of the squared Frobenius norm of the error with respect to time is given by:

$$\begin{aligned}
\frac{1}{2}\frac{d}{dt}\|Y - A\|_F^2 &= \langle \dot{Y} - \dot{A}, Y - A \rangle_F \\
&= \langle P(Y)F(Y) - F(A), Y - A \rangle_F \\
&= \langle P(Y)F(Y) - F(Y) + F(Y) - F(A), Y - A \rangle_F \\
&= \langle F(Y) - F(A), Y - A \rangle_F + \langle P(Y)F(Y) - F(Y), Y - A \rangle_F.
\end{aligned}$$

We use the definition of the orthogonal complement projector $P^{\perp}(Y)Z := Z - P(Y)Z$. This allows us to write the second term as $P(Y)F(Y) - F(Y) = -P^{\perp}(Y)F(Y)$. The expression becomes:

$$\frac{1}{2}\frac{d}{dt}\|Y - A\|_F^2 = \langle F(Y) - F(A), Y - A\rangle_F - \langle P^{\perp}(Y)F(Y), Y - A\rangle_F. \tag{3.19}$$

For the first term, we apply Assumption 1.3 (Lipschitz continuity) and obtain

$$\langle Y - A, F(Y) - F(A)\rangle_F \leq L\|Y - A\|_F^2. \tag{3.20}$$

For the second term, we apply the Cauchy–Schwarz inequality and Assumption 3.11 (Tangentiality). We note that $\|P^{\perp}(Y)F(Y)\|_F \leq \epsilon$. Thus, it holds that

$$|\langle P^{\perp}(Y)F(Y), Y - A\rangle_F| \leq \|P^{\perp}(Y)F(Y)\|_F \cdot \|Y - A\|_F \leq \epsilon\|Y - A\|_F. \tag{3.21}$$

Combining these bounds, we obtain the following differential inequality for $u(t) := \|Y(t) - A(t)\|_F$:

$$\frac{1}{2}\frac{d}{dt}\|Y - A\|_F^2 \leq L\|Y - A\|_F^2 + \epsilon\|Y - A\|_F. \tag{3.22}$$

Using the identity $\frac{1}{2}\frac{d}{dt}\|X\|_F^2 = \|X\|_F \frac{d}{dt}\|X\|_F$, we simplify this to

$$\frac{du}{dt} \leq Lu(t) + \epsilon. \tag{3.23}$$

Applying Gronwall's Lemma (3.5) to this differential inequality with the initial condition $u(0) \leq \delta$, we obtain

$$\|Y(t) - A(t)\|_F \leq e^{Lt}\|Y(0) - A(0)\|_F + \int_0^t e^{L(t-s)}\epsilon ds. \tag{3.24}$$

Since $\|Y(0) - A(0)\|_F \leq \delta$, we get

$$\|Y(t) - A(t)\|_F \leq e^{Lt}\delta + \epsilon\int_0^t e^{L(t-s)}ds. \tag{3.25}$$

Evaluating the integral in the second term yields

$$\int_0^t e^{L(t-s)}ds = e^{Lt}\int_0^t e^{-Ls}ds = e^{Lt}[-\frac{1}{L}e^{-Ls}]_0^t = \frac{e^{Lt} - 1}{L}, \tag{3.26}$$

which can be bounded. This completes the proof. □

Now we can show that Assumption 1.5 is weaker then Assumption 3.11. First, we show that Assumption 3.11 implies Assumption 1.5. Suppose that F satisfies Assumption 3.11. Since the intitial value is bounded by

$$\|\Phi_F^0(Y) - \Phi_{\mathrm{Pr}F}^0(Y)\|_F = 0. \tag{3.27}$$

Applying the error bound from Lemma 3.4, there exists an $h_0 > 0$ such that for all $0 \leq h \leq h_0$:

$$\|\Phi_F^h(Y) - \Phi_{\mathrm{Pr}F}^h(Y)\|_F \leq \tilde{\epsilon}\int_0^h e^{Ls}ds \leq e^{Lh}h\tilde{\epsilon}. \tag{3.28}$$

Here, $\Phi_{\mathrm{Pr}F}^h(Y)$ denotes the flow generated by the projection of the vector field. Since the projected flow $\Phi_{\mathrm{Pr}F}^h(Y)$ is by definition contained within the manifold M_r ($\Phi_{\mathrm{Pr}F}^h(Y) \in M_r$), it is itself a rank-$r$ matrix. This implies that the best rank-r approximation of $\Phi_F^h(Y)$ cannot be worse than the approximation provided by $\Phi_{\mathrm{Pr}F}^h(Y)$

$$\|\Phi_F^h(Y) - [[\Phi_F^h(Y)]]_r\|_F \leq \|\Phi_F^h(Y) - \Phi_{\mathrm{Pr}F}^h(Y)\|_F \leq e^{Lh}h\tilde{\epsilon}. \tag{3.29}$$

Hence, Assumption 1.5 is satisfied with $C_M = e^{Lh_0}$ and $\epsilon = \tilde{\epsilon}$. This demonstrates that if the vector field is nearly tangential to the manifold, then the exact flow of the system for short times is well approximated by a rank-r matrix.

However, it is important to emphasize that Assumption 1.5 does not necessarily imply Assumption 3.11, i.e., the existence of a good low-rank approximation does not require the tangent space projection error to be small. This will be already demonstrated in the following example, where the exact flow was excellently low-rank approximable, but the projection of the vector field led to a significantly larger error.

Example 3.6. Consider the differential equation for a matrix $Y(t) \in \mathbb{R}^{3\times 3}$:

$$\dot{Y}(t) = F(Y(t)) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & -10 \end{pmatrix} Y(t) + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 10^{-5}e^{-1} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
$$Y(0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 10^{-6} \end{pmatrix}$$

The initial matrix $Y(0)$ here has rank 2. Our goal is a rank-2 approximation.

1. Calculation of the Exact Flow $\Phi_F^h(Y)$ and its Best Rank-2 Approximation. The exact flow $\Phi_F^h(Y)$ is given in the [**1**] as:

$$\Phi_F^h(Y) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 10^{-6}(e^{10h} - 1) & 0 \\ 0 & 0 & 10^{-6}e^{-10h} \end{pmatrix}$$

We calculate $\Phi_F^1(Y)$ for a time step $h = 1$:

$$\Phi_F^1(Y) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 10^{-6}(e^{10\cdot 1} - 1) & 0 \\ 0 & 0 & 10^{-6}e^{-10\cdot 1} \end{pmatrix}$$

Using the numerical values $e^{10} \approx 22026.46579$ and $e^{-10} \approx 4.5399929 \times 10^{-5}$:

$$10^{-6}(e^{10} - 1) = 10^{-6}(22026.46579 - 1) = 10^{-6}(22025.46579) \approx 0.02202546579$$
$$10^{-6}e^{-10} = 10^{-6}(4.5399929 \times 10^{-5}) \approx 4.5399929 \times 10^{-11}$$

Thus, the numerical value of the exact flow at $h = 1$ is approximately:

$$\Phi_F^1(Y) \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.02202546579 & 0 \\ 0 & 0 & 4.5399929 \times 10^{-11} \end{pmatrix}$$

Now we determine the best rank-2 approximation $[[\Phi_F^1(Y)]]_2$. Since $\Phi_F^1(Y)$ is a diagonal matrix, its singular values are the absolute values of its diagonal elements:

$$\sigma_1 = |1| = 1$$
$$\sigma_2 = |0.02202546579| = 0.02202546579$$
$$\sigma_3 = |4.5399929 \times 10^{-11}| = 4.5399929 \times 10^{-11}$$

The two largest singular values are σ_1 and σ_2. For the best rank-2 approximation, we keep the two largest singular values and set the smallest one (σ_3) to zero, which corresponds to setting the associated diagonal element to zero:

$$[[\Phi_F^1(Y)]]_2 \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.02202546579 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The error of the best rank-2 approximation is given by the Frobenius norm of the difference matrix:

$$\begin{aligned} \left\|\Phi_F^1(Y) - [[\Phi_F^1(Y)]]_2\right\|_F &= \left\| \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.02202546579 & 0 \\ 0 & 0 & 4.5399929 \times 10^{-11} \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.02202546579 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\|_F \\ &= \left\| \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4.5399929 \times 10^{-11} \end{pmatrix} \right\|_F \\ &= \sqrt{(4.5399929 \times 10^{-11})^2} \\ &\approx 4.54 \times 10^{-11} \end{aligned}$$

This value is extremely small and is of the same order of magnitude as 1.24×10^{-10} stated in the [**1**], confirming that the exact flow can be excellently approximated by a rank-2 matrix.

2. Calculation of the Error of the Tangent Space Projection Method. We now compare the exact flow $\Phi_F^1(Y)$ with the flow generated by the tangent space projection method (Dirac-Frenkel principle, here denoted as $\Phi_{PF}^h(Y)$). The formula for this projected flow is given in the [**1**] as:

$$\Phi_{PF}^h(Y) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 10^{-6}e^{-10h} \end{pmatrix}$$

For $h = 1$, we get:

$$\Phi_{PF}^1(Y) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 10^{-6}e^{-10} \end{pmatrix} \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4.5399929 \times 10^{-11} \end{pmatrix}$$

The error between the exact flow and the projected flow is given by the Frobenius norm of the difference matrix:

$$\begin{aligned} \left\|\Phi_F^1(Y) - \Phi_{PF}^1(Y)\right\|_F &= \left\| \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.02202546579 & 0 \\ 0 & 0 & 4.5399929 \times 10^{-11} \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4.5399929 \times 10^{-11} \end{pmatrix} \right\|_F \\ &= \left\| \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.02202546579 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\|_F \\ &= \sqrt{(0.02202546579)^2} \\ &\approx 0.02202546579 \end{aligned}$$

This value is approximately 0.022, which is significantly larger than the stated value of ≥ 0.008 in [**1**] and lies in a completely different order of magnitude than the error of the best rank-2 approximation (10^{-11}).

This demonstrates that the property of low-rank approximability of the solution (the flow) does not necessarily imply that the vector field $F(Y)$ itself is strongly tangential to the low-rank manifold. If the "non-tangential" component of the vector field is large, forcing the solution onto the manifold by projecting the vector field introduces a noticeable error, even if the exact solution is inherently low-rank.

With this framework established, we will now return to the static approximation problem and discuss **Stage B**, beginning with the Nyström method for constructing a complete factorization from the subspace obtained in Stage A.

3.4. **Stage B: Construction of standard factorizations.** With the geometric framework for handling dynamics established, we now return to the second part of our two-stage approximation process. **Stage A** provided an algorithm to identify a low-dimensional subspace that captures the action of a matrix A. **Stage B**, which we detail now, uses this subspace information to construct an explicit low-rank factorization. For this purpose, we employ the **generalized Nyström method**, a technique that is both computationally efficient and numerically stable.

The classical Nyström method is a fast algorithm for matrix approximation, but its application is restricted to positive semidefinite matrices. The work of Yuji Nakatsukasa [**19**] introduces a generalization of the Nyström method that is applicable to general matrices. This method exhibits several advantages,

(1) It achieves a near-optimal approximation quality, which is comparable to that of competing methods.
(2) The computational cost is near-optimal, specifically $O(nm \log n + r^3)$ for dense matrices, with small hidden constants.
(3) Most important, it can be implemented in a numerically stable fashion, despite the presence of an ill-conditioned pseudoinverse.

respectively [**19**].

3.4.1. *Generalized Nyström Method.* The generalized Nyström method constructs a low-rank approximation by creating a sketch of both the columns and rows of a matrix. Given a matrix $Z \in \mathbb{R}^{m\times n}$ and integer oversampling parameters $p, l \in \mathbb{N}$, we draw two random test matrices first $\Omega \in \mathbb{R}^{n\times(r+p)}$ to sample the columns of Z and second $\Psi \in \mathbb{R}^{m\times(r+l)}$ to sample the rows of Z.

The column sketch is given by $Z\Omega$, and the row sketch is given by $\Psi^T Z$. The approximation is formed by projecting Z onto the subspace spanned by the column sketch, span($Z\Omega$), using an oblique projection

defined by the row sketch. The resulting rank-r approximation is

$$N(Z) := \left[Z\Omega(\Psi^T Z\Omega)^\dagger \Psi^T Z\right]_r, \tag{3.30}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse.

A straightforward application of the formula above encounters practical difficulty since the matrix $\Psi^T Z\Omega$ is often ill-conditioned and the use of the pseudoinverse $(\cdot)^\dagger$ may therefore lead to significant numerical errors.

To overcome this instability, we can use a more robust, equivalent formulation. The key idea is to first compute an orthonormal basis for the sketched subspace. Let $Q \in \mathbb{R}^{m\times n}$ be a matrix whose columns form an orthonormal basis for the range of the column sketch $Z\Omega$, typically computed via a QR factorization, $Z\Omega = QR$. Since $span(Q) = span(Z\Omega)$, we can replace $Z\Omega$ with the well-conditioned matrix Q in the projection formula

$$N(Z) = Q[[(\Psi^T Q)^\dagger \Psi^T Z]]_r. \tag{3.31}$$

This formulation is mathematically equivalent to the original but numerically far more stable, as the columns of Q are orthogonal, leading to a much better-conditioned matrix $\Psi^T Q$. This QR-based approach is therefore adopted as the standard method for our subsequent experiments.

Even with the QR factorization, the core matrix (e.g., $\Psi^T Q$) can have very small singular values due to near-linear dependencies introduced by the random sampling. To guard against this, the pseudoinverse is often implemented as a truncated pseudoinverse, also known as the ϵ-pseudoinverse.

If the SVD of the core matrix $C = \Psi^T Q$ is given by $C = U\Sigma V^T$, where Σ is a diagonal matrix of singular values, we can partition it as

$$C = [U_1, U_2] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} [V_1, V_2]^T, \tag{3.32}$$

where Σ_1 contains all singular values larger than ϵ. The ϵ-pseudoinverse is then computed by inverting only these singular values

$$C_\epsilon^\dagger = V_1 \Sigma_1^{-1} U_1^T. \tag{3.33}$$

According to [**19**], a suitable choice for the threshold is $\epsilon = O(u\|A\|)$, where u is the unit roundoff (machine precision, e.g., $u \approx 10^{-16}$). This step effectively filters out numerical noise and ensures the stability of the final computation.

3.4.2. *Error Analysis.* A common question is how well the random Nyström approximation $N(Z)$ performs compared to the optimal rank-r approximation $[[Z]]_r$. The quality of a random approximation map $R : \mathbb{R}^{m\times n} \to \mathcal{M}_r$ can be measured through its error moments. We say that R satisfies the **moment assumption** for $q \geq 1$ if there exists a constant C_R, independent of Z, such that for any matrix $Z \in \mathbb{R}^{m\times n}$:

$$\|R(Z) - Z\|_{L_q} \leq C_R \|Z - [[Z]]_r\|_F, \tag{3.34}$$

where $\|\cdot\|_{L_q} = (\mathbb{E}[\|\cdot\|_F^q])^{1/q}$. This inequality states that the expected error of the random approximation is proportional to the smallest possible error, which is the error of the best rank-r approximation. As will be shown, the generalized Nyström method (using Gaussian random matrices) satisfies the moment assumption. To prove the moment assumption we need the following theorem.

Theorem 3.7 (Law of Total Expectation)**.** *Let X be a random variable with expected value $E(X)$ and let Y be any random variable defined on the same probability space. Then, the expected value of the conditional expectation of X given Y is the same as the expected value of X:*

$$E(X) = E[E(X|Y)]. \tag{3.35}$$

Theorem 3.8. *Suppose that $\Omega \in \mathbb{R}^{n\times(r+p)}$ and $\Psi \in \mathbb{R}^{m\times(r+p+\ell)}$ are independent standard Gaussian matrices with $p, \ell \geq 4$. Setting $q = \min\{p, \ell\}$, it holds for $Z \in \mathbb{R}^{m\times n}$ that*

$$\|\mathcal{N}(Z) - Z\|_{L^q} = (\mathbb{E}[\|\mathcal{N}(Z) - Z\|_F^q])^{1/q} \leq C_N \|Z - [[Z]]_r\|_F, \tag{3.36}$$

with

$$C_N = 1 + 2\sqrt{(1 + r + p)(1 + r)}. \tag{3.37}$$

Proof. By the triangle inequality,

$$\begin{aligned}\|\mathcal{N}(Z) - Z\|_{L^q} &\leq \left\|[[Z(\Psi^T Z)^\dagger \Psi^T Z]]_r - Z(\Psi^T Z)^\dagger \Psi^T Z\right\|_{L^q} \\ &\quad + \left\|Z(\Psi^T Z)^\dagger \Psi^T Z - Z\right\|_{L^q} \\ &\leq \left\|[[Z]]_r - Z(\Psi^T Z)^\dagger \Psi^T Z\right\|_{L^q} + \left\|Z(\Psi^T Z)^\dagger \Psi^T Z - Z\right\|_{L^q} \\ &\leq \|[[Z]]_r - Z\|_{L^q} + 2\left\|Z(\Psi^T Z)^\dagger \Psi^T Z - Z\right\|_{L^q}.\end{aligned}$$

To bound the second term, we follow the proof of [12, Theorem 11]. A detailed version of this proof can be found in the Appendix, see Section A.1.

□

3.5. **An idealised projection method.** Assume that $y_n \in M$. A single step from y_n to y_{n+1} is defined as follows

- **Step 1:** Compute an intermediate value $\tilde{y}_{n+1} = \Phi_h(y_n)$, where Φ_h is an arbitrary one-step numerical method applied to the ordinary differential equation $\dot{y} = f(y)$.
- **Step 2:** Project the intermediate value $\tilde{y}_{n+1}$ onto the manifold M to obtain the next point $y_{n+1} \in M$.

Following [**1**] and [**16**], a rank-r approximation to the solution $A((i+1)h)$ can be obtained by combining the exact flow with a subsequent rank-r truncation

$$Y_{i+1} = [\Phi_F^h(Y_i)]_r, \quad Y_0 = [A_0]_r. \tag{3.38}$$

This scheme is referred to as *idealized*, since the exact flow Φ_F^h cannot be evaluated in practice and needs to be approximated. By replacing the rank-r truncation with the generalized Nyström method, we obtain the so-called idealized randomized projection method

$$Y_{i+1} = \mathcal{N}_i(\Phi_F^h(Y_i)) = \left[\Phi_F^h(Y_i)\Omega_i(\Psi_i^T \Phi_F^h(Y_i)\Omega_i)^\dagger \Psi_i^T \Phi_F^h(Y_i)\right]_r, \quad Y_0 = \mathcal{N}_0(A_0). \tag{3.39}$$

Here, the index i indicates that in each time step the generalized Nyström approximation is applied with independent test matrices $\Omega_i \in \mathbb{R}^{n\times(r+p)}$ and $\Psi_i \in \mathbb{R}^{m\times(r+p+l)}$. It satisfies the following error estimate without a restriction.

Theorem 3.9 (Gloabal error)**.** *With the assumptions stated in Section and assuming $\|[[A_0]]_r - A_0\|_F \leq \delta$ holds for the initial data, the method (10) with independent standard Gaussian matrices $\Omega_i \in \mathbb{R}^{n\times(r+p)}$, $\Psi_i \in \mathbb{R}^{m\times(r+p+\ell)}$ and oversampling parameters $p, \ell \geq 4$ satisfies the error estimate*

$$\|Y_N - A(Nh)\|_{L_q} \leq C(\delta + \epsilon) \tag{3.40}$$

for $q = \min\{p, \ell\}$ on a finite time-interval $0 \leq Nh \leq T$ for every $0 < h \leq h_0$. The constant C only depends on L, T, h_0, C_M, and C_N.

First, we prove a lemma that establishes the Lipschitz continuity of the flow, which is a direct consequence of the Lipschitz continuity of the underlying vector field.

Lemma 3.10 (Lipschitz Continuity of the Flow)**.** *Let F be Lipschitz continuous on a domain $D \subseteq \mathbb{R}^{d\times m}$ with a Lipschitz constant $L \geq 0$, i.e.,*

$$\|F(X) - F(Y)\|_F \leq L\|X - Y\|_F \quad \forall X, Y \in D. \tag{3.41}$$

Then, the corresponding flow Φ_F^t is also Lipschitz continuous for any $t \geq 0$, and satisfies the inequality:

$$\|\Phi_F^t(X) - \Phi_F^t(Y)\|_F \leq e^{Lt}\|X - Y\|_F. \tag{3.42}$$

Proof. We can express the solutions for two different starting points, X and Y, in their integral forms

$$\Phi_F^t(X) = X + \int_0^t F(\Phi_F^s(X))\, ds$$
$$\Phi_F^t(Y) = Y + \int_0^t F(\Phi_F^s(Y))\, ds.$$

We consider the norm of their difference

$$\|\Phi_F^t(X) - \Phi_F^t(Y)\|_F = \left\|(X - Y) + \int_0^t (F(\Phi_F^s(X)) - F(\Phi_F^s(Y)))\, ds\right\|_F \tag{3.43}$$

Applying the triangel inequality and the Lipschitz continuity of F, we derive

$$\left\|(X - Y) + \int_0^t (F(\Phi_F^s(X)) - F(\Phi_F^s(Y)))\, ds\right\|_F \leq \|X - Y\|_F + \int_0^t L\|\Phi_F^s(X) - \Phi_F^s(Y)\|_F\, ds. \tag{3.44}$$

Now we define the function $d(t) := \|\Phi_F^t(X) - \Phi_F^t(Y)\|_F$. The inequality then becomes

$$d(t) \leq d(0) + L \int_0^t d(s)\, ds. \tag{3.45}$$

Applying Gronwall's Lemma directly yields the desired result

$$d(t) \leq d(0)e^{Lt}, \tag{3.46}$$

which, when written out, is precisely

$$\|\Phi_F^t(X) - \Phi_F^t(Y)\|_F \leq \|X - Y\|_F e^{Lt}. \tag{3.47}$$

□

Now we can proof the bound of the global error (Theorem 3.9).

Proof. The objective is to bound the global error $\|Y_N - A(Nh)\|_{L_q}$, where $A(Nh) = \Phi_F^{Nh}(A_0)$ is the exact solution.

The main idea of the proof is to express the global error as a sum of propagated local errors. We achieve this using a telescoping sum.

Derivation of the Telescoping Sum. The total error is the difference between the final numerical solution, Y_N, and the final exact solution, $\Phi_F^{Nh}(A_0)$. We can decompose this error by adding and subtracting a sequence of intermediate terms that cancel each other out.

First, we split the error into the error from the initial condition and the accumulated error over all steps:

$$Y_N - \Phi_F^{Nh}(A_0) = \underbrace{(Y_N - \Phi_F^{Nh}(Y_0))}_{\text{Accumulated Error}} + \underbrace{(\Phi_F^{Nh}(Y_0) - \Phi_F^{Nh}(A_0))}_{\text{Initial Error}} \tag{3.48}$$

Now, we rewrite the accumulated error term, $Y_N - \Phi_F^{Nh}(Y_0)$, as a sum. Recognizing that $Y_N = \Phi_F^0(Y_N)$, we can express this term as

$$\begin{aligned} Y_N - \Phi_F^{Nh}(Y_0) &= \Phi_F^0(Y_N) - \Phi_F^{Nh}(Y_0) \\ &= \left(\Phi_F^0(Y_N) - \Phi_F^h(Y_{N-1})\right) \\ &\quad + \left(\Phi_F^h(Y_{N-1}) - \Phi_F^{2h}(Y_{N-2})\right) \\ &\quad + \dots \\ &\quad + \left(\Phi_F^{(N-1)h}(Y_1) - \Phi_F^{Nh}(Y_0)\right) \end{aligned}$$

The second term in each parenthesis cancels with the first term in the next, which is the definition of a telescoping sum. This can be written compactly as

$$Y_N - \Phi_F^{Nh}(Y_0) = \sum_{i=1}^{N} \left(\Phi_F^{(N-i)h}(Y_i) - \Phi_F^{(N-i+1)h}(Y_{i-1})\right). \tag{3.49}$$

To proceed, we use the semigroup property of the flow, $\Phi_F^{t+s}(X) = \Phi_F^t(\Phi_F^s(X))$, to rewrite the second term in the sum

$$\Phi_F^{(N-i+1)h}(Y_{i-1}) = \Phi_F^{(N-i)h+h}(Y_{i-1}) = \Phi_F^{(N-i)h}(\Phi_F^h(Y_{i-1})). \tag{3.50}$$

This transforms our sum into a series of comparisons between the propagated numerical solution Y_i and the propagated exact solution from the previous step $\Phi_F^h(Y_{i-1})$.

Applying the Triangle Inequality. Substituting this sum back into our original expression and applying the triangle inequality yields

$$\|Y_N - A(Nh)\|_{L_q} \leq \underbrace{\|\Phi_F^{Nh}(Y_0) - \Phi_F^{Nh}(A_0)\|_{L_q}}_{E_0} + \sum_{i=1}^{N} \underbrace{\|\Phi_F^{(N-i)h}(Y_i) - \Phi_F^{(N-i)h}(\Phi_F^h(Y_{i-1}))\|_{L_q}}_{E_i} \tag{3.51}$$

We now bound the terms E_0 and E_i individually.

Bounding the Initial Error E_0. By applying the Lipschitz continuity of the flow and the assumption that $\|Y_0 - A_0\|_F \leq \delta$, we can bound (3.43)

$$E_0 \leq e^{LNh}\|Y_0 - A_0\|_{L_q} \leq e^{LNh}\delta. \tag{3.52}$$

Bounding the Propagated Local Error E_i. Next, we bound the propagated local error terms E_i for $i = 1, \ldots, N$, which are defined as

$$E_i = \|\Phi_F^{(N-i)h}(Y_i) - \Phi_F^{(N-i)h}(\Phi_F^h(Y_{i-1}))\|_{L_q}. \tag{3.53}$$

By applying the law of total expectation, we can condition the expectation on the previous state Y_{i-1}

$$E_i = \left(\mathbb{E}\left[\mathbb{E}\left[\|\Phi_F^{(N-i)h}(Y_i) - \Phi_F^{(N-i)h}(\Phi_F^h(Y_{i-1}))\|_F^q \mid Y_{i-1}\right]\right]\right)^{1/q}. \tag{3.54}$$

The Lipschitz continuity of the flow allows us to pull the evolution outside the expectation, scaled by an exponential factor

$$E_i \leq e^{Lh(N-i)} \left(\mathbb{E}\left[\mathbb{E}\left[\|Y_i - \Phi_F^h(Y_{i-1})\|_F^q \mid Y_{i-1}\right]\right]\right)^{1/q}. \tag{3.55}$$

Substituting and applying the moment assumption on the randomized projection yields

$$E_i \leq e^{Lh(N-i)} C_N \left(\mathbb{E}\left[\|\Phi_F^h(Y_{i-1}) - [\Phi_F^h(Y_{i-1})]_r\|_F^q\right]\right)^{1/q}. \tag{3.56}$$

Invoking the low-rank approximability of the flow (Assumption 3) gives the final bound for the local error term

$$E_i \leq e^{Lh(N-i)} C_N (C_M h\epsilon). \tag{3.57}$$

Finally, we sum the individual error bounds

$$\|Y_N - A(Nh)\|_{L_q} \leq E_0 + \sum_{i=1}^{N} E_i \leq e^{LNh}\delta + \sum_{i=1}^{N} e^{Lh(N-i)} C_N C_M h\epsilon. \tag{3.58}$$

For a fixed final time $T = Nh$, the exponential terms are bounded by e^{LT}. The summation can likewise be bounded by a constant times T. This leads to the final error estimate:

$$\|Y_N - A(Nh)\|_{L_q} \leq C(\delta + \epsilon) \tag{3.59}$$

□

3.6. **Computational Efficiency of Dense vs. Structured Sketching Methods.** While dense random matrices provide a robust method for randomized sketching, their application via matrix multiplication creates a computational bottleneck. Multiplyying a square matrix of dimension n with gaussian matrix $\Omega \in \mathbb{R}^{m\times n}$ we obtain a complexity of $O(mn^2)$. This section explores a more efficient approach using structured random matrices, specifically the Subsampled Randomized Fourier Transform (SRFT), which significantly reduces this complexity. Here, we consider a data matrix $A \in \mathbb{R}^{m\times n}$ and aim to compute a sketch $Y \in \mathbb{C}^{m\times r}$ that captures its dominant actions, where r is the target rank. The SRFT achieves this by employing the Subsampled Fast Fourier Transform , a efficient algorithm that uses a "divide and conquer" strategy to compute the Discrete Fourier Transform [**24**].

Definition 3.11 (Subsampled Randomized Fourier Transformation (SRFT))**.** An $n \times l$ SRFT matrix is a structured random matrix defined by the form

$$\Omega = \sqrt{\frac{n}{l}} DFR. \tag{3.60}$$

With the components being defined as follows:

- **D** is an $n \times n$ diagonal matrix whose entries are independent random variables uniformly distributed on the complex unit circle.
- **F** is the $n \times n$ unitary discrete Fourier transform (DFT), whose entries take the values $f_{pq} = n^{-1/2} e^{-2\pi i(p-1)(q-1)/n}$ for $p, q = 1, 2, \ldots, n$.
- **R** is an $n \times l$ matrix that samples l coordinates from n uniformly at random, i.e., its l columns are drawn randomly without replacement from the columns of the $n \times n$ identity matrix.

To empirically validate the theoretical performance advantages, a numerical benchmark was designed. The data matrix $A \in \mathbb{R}^{m\times n}$ was a standard Gaussian random matrix with entries drawn independently from $\mathcal{N}(0, 1)$.

The two compared sketching methods are:

(1) **Dense Randomized Sketching:** Computes the sketch by multiplying A with a dense Gaussian random matrix $G \in \mathbb{R}^{n\times r}$.
(2) **Structured Randomized Sketching:** Applies an SRFT matrix $\Omega \in \mathbb{C}^{n\times r}$ to A using its associated fast algorithm.

In the first experiment, we set the dimensions to $n = 8192$, $m = 1024$, and the target rank to $r = 2048$. The runtimes for computing the sketch $Y = A\Omega$ were averaged over 10 runs. As illustrated in Figure **??**, the performance difference is small. The SRFT-based sketch, computed via the FFT, is slightly faster than the standard multiplication with a dense Gaussian matrix. T

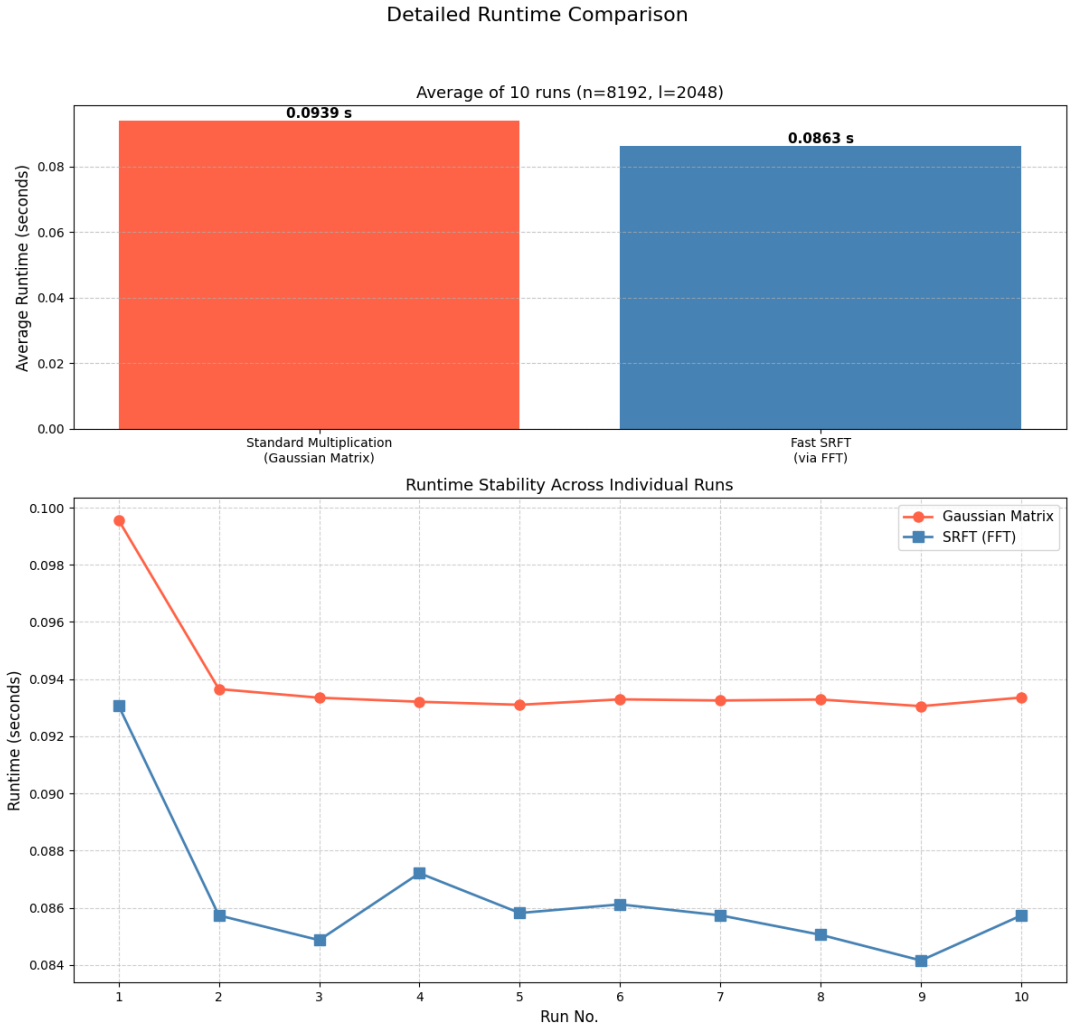

FIGURE 5. Performance comparison for large dimensions ($n = 8192, r = 2048$) on a random data matrix A. The benchmark compares a standard multiplication with a dense Gaussian matrix against the application of an SRFT matrix.

To investigate the scaling behavior, the dimensions were increased to $n = 24576$ and $r = 6144$. The results from this larger test, shown in Figure 6, are more pronounced. The runtime for the dense Gaussian multiplication increased drastically, as predicted by its linear dependency on the now much larger factor r. In contrast, the runtime for the SRFT application grew much more modestly.

The demonstrated efficiency of the SRFT motivates its use in an accelerated version of the basic Randomized Range Finder. By replacing the dense Gaussian matrix with a structured SRFT matrix, we obtain.

It is noteworthy that the literature often refers to a more specialized *subsampled FFT*.The only difference lies in Step 2. Such an optimized algorithm does not compute the full n-point Fourier Transform but calculates only the r required output coordinates directly. This can further reduce the theoretical complexity to $O(mn \log r)$. For the practical experiments in this work, however, the standard FFT approach was chosen for its straightforward implementation.

This high efficiency is paramount, as the range-finding step forms the computational core of each time step in the Randomized Low-Rank Runge-Kutta method discussed in Chapter 4.

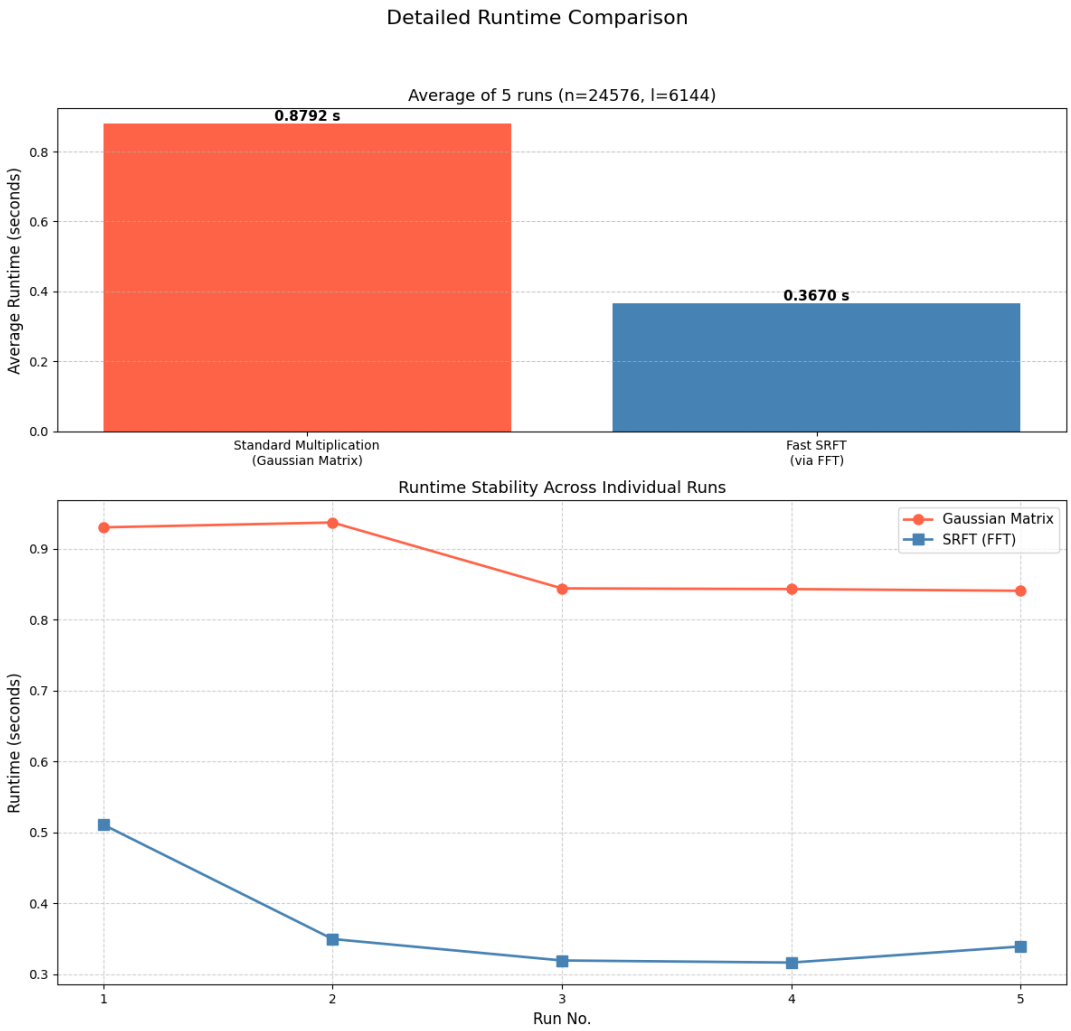

FIGURE 6. Performance comparison for very large dimensions ($n = 24576, r = 6144$). The performance gap widens significantly, highlighting the superior scalability of the SRFT over the dense sketching approach.

Algorithm 2: Fast Randomized Range Finder

Input: An $m \times n$ matrix A and an integer ℓ. **Output:** An $m \times \ell$ orthonormal matrix Q.

(1) Generate an $n \times \ell$ SRFT test matrix Ω (as defined in (3.40).
(2) Compute the $m \times \ell$ sketch matrix $Y = A\Omega$ using a Fast Fourier Transform (FFT).
(3) Construct an $m \times \ell$ matrix Q whose columns form an orthonormal basis for the range of Y, for example, using a QR factorization $Y = QR$.

FIGURE 7. The Fast Randomized Range Finder algorithm, which uses a structured SRFT matrix to accelerate the sketching step.

4. Randomized Low-Rank Runge-Kutta (RLRK) Method

Having established the foundational principles of both Runge-Kutta methods and randomized low-rank approximation in the preceding chapters, we now proceed to their synthesis. This chapter introduces the **Randomized Low-Rank Runge-Kutta (RLRK) method**, an effective numerical integrator designed for high-dimensional matrix differential equations. The method was first proposed by Lam, Ceruti, and Kressner in [**1**].

4.1. **Algorithm and Implementation.** A direct application of a conventional Runge-Kutta scheme is limited by high dimensionality. As the method progresses through its internal stages, the summation over previous steps causes an increase in the rank of the intermediate solutions. This phenomenon quickly negates the advantages of a pre-existing low-rank structure, leading to high computational costs.

The RLRK method circumvents this issue by integrating the rank-reduction process directly into the time-stepping procedure. The idea is to apply a randomized projection (e.g. the generalized Nyström method $\mathcal{N}$) to constrain the rank after each conceptual stage is formed. The resulting scheme, for an approximation Y_i at the beginning of a time step, is defined as

$$Z_j = Y_i + h\sum_{l=1}^{j-1} a_{jl} F(\mathcal{N}_l(Z_l)), \quad \text{for } j = 1, \dots, s, \tag{4.1}$$

$$Y_{i+1} = \mathcal{N}_{s+1}\left(Y_i + h\sum_{j=1}^{s} b_j F(\mathcal{N}_j(Z_j))\right). \tag{4.2}$$

The index of $\mathcal{N}$ signifies that the random matrices Ω, Ψ are drawn independently not only at each stage within a given time step but also across separate time steps.

A examination of the intermediate matrix Z_j reveals that its rank exhibits linear growth with respect to the stage index j. This can be understood by considering the subadditivity property of the matrix rank. Since $\operatorname{rank}(Y_i) = r$, we assume that the application of the function F to a rank-r matrix yields a result whose rank can be bounded by r_F, i.e., $\operatorname{rank}(F(\mathcal{N}_l(Z_l))) \approx r_F$. The rank of Z_j is then bounded by the sum of the ranks of its component terms

$$\operatorname{rank}(Z_j) = \operatorname{rank}\left(Y_i + h\sum_{l=1}^{j-1} a_{jl} F(\mathcal{N}_l(Z_l))\right) \leq \operatorname{rank}(Y_i) + \sum_{l=1}^{j-1} \operatorname{rank}(F(\mathcal{N}_l(Z_l))) \leq r + (j-1)r_F. \tag{4.3}$$

While this linear growth appears to present a computational obstacle, the practical implementation of the RLRK method circumvents this issue.

The algorithm never explicitly constructs the potentially high-rank matrix Z_j. As established in Chapter 3, the Nyström method makes the explicit formation of the full matrix unnecessary, as it only requires low-dimensional sketches. Leveraging the linearity of the sketching process, we can compute the sketches of Z_j directly from the sketches of its constituent terms. This allows the entire algorithm to be reformulated to operate exclusively on small sketch matrices

$$Z_j\Omega_j = Y_i\Omega_j + h\sum_{l=1}^{j-1} a_{jl}\left(F(\mathcal{N}_l(Z_l))\Omega_j\right), \tag{4.4}$$

$$\Psi_j^T Z_j = \Psi_j^T Y_i + h\sum_{l=1}^{j-1} a_{jl}\left(\Psi_j^T F(\mathcal{N}_l(Z_l))\right). \tag{4.5}$$

The evaluation of $N_j(Z_j)$ via expression (3.31) requires only the terms $\Psi_j^T Z_j$, Ψ_j^T, and $Z_j\Omega_j$ (or rather the orthogonal factor from its QR decomposition). Algorithm 4.1 outlines the pseudocode for the randomized low-rank Runge-Kutta method. This algorithm largely follows the procedure described in (4.4 and 4.5), with the difference being the precomputation of the sketches for $F([[\hat{Z}_j(\Psi_j^T\hat{Z}_j)^\dagger\tilde{Z}_j]]_r)$, which are necessary for the subsequent stages.

4.1.1. *Computational Cost.* Having detailed the procedural steps of the RLRK algorithm, we now provide a rigorous analysis of the computational cost of a single time step. To facilitate the analysis, we establish several key definitions and assumptions. simplify the analysis, we assume the matrices are square, i.e., $m = n$, and that the oversampling parameters are proportional to the target rank, $p, \ell = \mathcal{O}(r)$, where $r \ll n$. Furthermore, we denote the cost of multiplying a vector of length n with a sketching matrix Ω or Ψ as c_n. In the worst-case scenario, when using unstructured dense random matrices (e.g., Gaussian matrices), this cost is $c_n = \mathcal{O}(nr)$. However, as discussed in Section 3.5, this cost can be significantly reduced by using structured random matrices. Finally, the cost of applying the function F to a rank-r matrix is denoted by c_F, and the rank of the resulting matrix is r_F, which is assumed to be at least as large as the target rank, i.e., $r_F \geq r$. In the implementation, every large $n \times n$ matrix that occurs in the algorithm is represented in a factored form, $U\Sigma V^T$. In this representation, $U, V \in \mathbb{R}^{n\times r}$ are tall matrices, and $\Sigma \in \mathbb{R}^{r\times r}$ is a small square matrix. It is important to note that the factors U and V are not necessarily kept orthonormal. This choice is deliberate and motivated by computational efficiency. Enforcing orthonormality at each internal stage (e.g., via a QR or SVD factorization) would

Algorithm 1 Randomized low-rank Runge-Kutta method with s stages

Input: Differential equation defined by F and initial condition $A_0 \in \mathbb{R}^{m\times n}$. Target rank r, oversampling parameters p, ℓ, step size $h > 0$, number of steps $N \geq 0$.
Output: Approximation $Y_N \in \mathcal{M}_r$ of $A(Nh)$.
Draw ind. random matrices $\Omega \in \mathbb{R}^{n\times(r+p)}$, $\Psi \in \mathbb{R}^{m\times(r+p+\ell)}$.
$\hat{Y}_0 \leftarrow A_0\Omega, \quad \tilde{Y}_0 \leftarrow \Psi^T A_0$
for $i = 0, \ldots, N-1$ **do**
  Draw ind. random matrices $\Omega_j \in \mathbb{R}^{n\times(r+p)}$, $\Psi_j \in \mathbb{R}^{m\times(r+p+\ell)}$ for $j = 1, 2, \ldots, (s+1)$.
  for $j = 1, \ldots, s$ **do**
    $\hat{Z}_j \leftarrow [[\hat{Y}_i(\Psi^T\hat{Y}_i)^\dagger\tilde{Y}_i]]_r\Omega_j + h\sum_{l=1}^{j-1} a_{jl}\hat{K}_{lj}$ ▷ Evaluate $[[\ldots]]_r$ using Eq. 3.31
    $\tilde{Z}_j \leftarrow \Psi_j^T[[\hat{Y}_i(\Psi^T\hat{Y}_i)^\dagger\tilde{Y}_i]]_r + h\sum_{l=1}^{j-1} a_{jl}\tilde{K}_{lj}$
    $F_j \leftarrow F([[\hat{Z}_j(\Psi_j^T\hat{Z}_j)^\dagger\tilde{Z}_j]]_r)$ ▷ Evaluate $[[\ldots]]_r$ using Eq. 3.31
    for $q = j+1, \ldots, s+1$ **do** ▷ Pre-compute sketches of F for subsequent stages
      $\hat{K}_{jq} \leftarrow F_j\Omega_q$
      $\tilde{K}_{jq} \leftarrow \Psi_q^T F_j$
    end for
  end for
  $\hat{Y}_{i+1} \leftarrow Y_i\Omega_{s+1} + h\sum_{j=1}^{s} b_j\hat{K}_{j,s+1}$
  $\tilde{Y}_{i+1} \leftarrow \Psi_{s+1}^T Y_i + h\sum_{j=1}^{s} b_j\tilde{K}_{j,s+1}$
  Set $\Psi \leftarrow \Psi_{s+1}$
end for
Return $Y_N = [[\hat{Y}_N(\Psi^T\hat{Y}_N)^\dagger\tilde{Y}_N]]_r$. ▷ Compute $[[\ldots]]_r$ using Eq. 3.31

be computationally intensive and prohibitive. This economic factored representation is sufficient for the algorithm's operations and avoids unnecessary computational overhead.

To determine the total cost of a full time step, we first analyze the complexity of a single, generic stage j within the inner loop of Algorithm 4.1. The cost of this stage can be decomposed into two primary components. First, we consider the computation of the stage input $\hat{Z}_j$, noting that the procedure for $\tilde{Z}_j$ is analogous. This step involves the summation of a rank-r approximation and $j-1$ previously computed sketch terms. The cost for forming the content of $\hat{Z}_j$ is thus $\mathcal{O}(nr^2 + jnr)$. Subsequently, computing its final factored, rank-r form via the Nyström method incurs an additional cost of $\mathcal{O}(nr^2)$.

Second, after applying the function F at a cost of c_F, the resulting rank-r_F matrix F_j must be sketched for the $s+1-j$ subsequent stages. As F_j is represented by its factors $U_F\Sigma_F V_F^T$, this sketch is computed efficiently as $\hat{K}_{jq} = U_F(\Sigma_F(V_F^T\Omega_q))$. The cost of this operation is dominated by multiplications involving the tall factor matrices, resulting in a complexity of $\mathcal{O}(r_F c_n + nr_F r)$ for each of the $s+1-j$ pre-computations.

Combining these components, the total complexity for the j-th stage is given by

$$\text{Cost(Stage } j) = \mathcal{O}\left(jnr + nr^2 + c_F + (s+1-j)(r_F c_n + nr_F r)\right). \tag{4.6}$$

To obtain the complexity of a full time step, we sum this cost over all s stages

$$C_{\text{total}} = \sum_{j=1}^{s}\left(jnr + nr^2 + c_F + (s+1-j)(r_F c_n + nr_F r)\right). \tag{4.7}$$

By separating the terms and applying the formula for the sum of the first s integers, $\sum_{j=1}^{s} j = \frac{s(s+1)}{2}$, the total cost is found to be

$$C_{\text{total}} = \mathcal{O}\left(s^2nr + snr^2 + sc_F + s^2(r_F c_n + nr_F r)\right). \tag{4.8}$$

Since $r_F \geq r$, the term $\mathcal{O}(s^2nr_F r)$ dominates $\mathcal{O}(s^2nr)$. The latter is therefore subsumed, yielding the final complexity for one full time step as

$$C_{\text{total}} = \mathcal{O}\left(s^2(r_F c_n + nr_F r) + snr^2 + sc_F\right). \tag{4.9}$$

4.1.2. *Computational advantages over Randomized SVD.* The choice of the generalized Nyström method as the rank-reduction tool is deliberate, as it avoids significant drawbacks associated with the popular Randomized SVD in the context of Runge-Kutta integrators. An advantage of the Nyström method is the **linearity** of its sketching operator. For matrices A and B, the sketch of a sum is the sum of the sketches ($\text{Sketch}(A+B) = \text{Sketch}(A) + \text{Sketch}(B)$). This property is beneficial, as it allows the algorithm

to only compute and store the small sketch matrices of the terms in the Runge-Kutta sum and add them afterwards. This approach is memory-efficient since only the small sketches need to be stored.

In contrast, the randomized SVD approximation is inherently **non-linear**. Its central step, the formation of an orthonormal basis, depends on the entirety of the matrix A. Consequently, the approximation of a sum is not equal to the sum of approximations ($\text{Approx}(A+B) \neq \text{Approx}(A) + \text{Approx}(B)$). This non-linearity requires one to first form the full sum $Z = Y + h\sum a_{jl}F_l$ by adding the large factor matrices, and only then apply the randomized SVD. This necessitates holding all factor matrices in memory simultaneously, leading to significantly higher memory requirements.

Furthermore, this non-linearity leads to a prohibitive computational cost. A sum of s matrices of rank r_F can have a rank up to $k = s \cdot r_F$. The dominant cost in the randomized SVD algorithm is the QR factorization of the sketch, which for a matrix with k columns is $\mathcal{O}(nk^2)$, respectively [**26**; **5**]. Substituting $k = s \cdot r_F$ yields a complexity of $\mathcal{O}(ns^2r_F^2)$. This cost, scaling quadratically with both the number of stages s and the rank r_F, is substantially higher than that of the Nyström-based approach, making randomized SVD ill-suited for this class of problems.

4.2. **Comparison to Projected Runge-Kutta method.** The Projected Runge-Kutta (PRK) method, introduced in [**3**], is closely related to our method (4.1). An iteration of the PRK method is given by

$$Z_j = Y_i + h\sum_{l=1}^{j-1} a_{jl}\text{Pr}(\mathcal{R}(Z_l))F(\mathcal{R}(Z_l)), \quad j = 1, \dots, s, \tag{4.10}$$

$$Y_{i+1} = \mathcal{R}\left(Y_i + h\sum_{j=1}^{s} b_j\text{Pr}(\mathcal{R}(Z_j))F(\mathcal{R}(Z_j))\right). \tag{4.11}$$

Here, $\mathcal{R}$ denotes a retraction to the manifold $\mathcal{M}_r$, for which the truncated SVD is a common choice. The tangent space projection Pr in (4.10) reduces the rank of $F(\mathcal{R}(Z_l))$ to $2r$, which can significantly lower the computational cost of the subsequent retraction $\mathcal{R}$. A single iteration of the PRK method (4.10) requires applying the retraction to the stage matrices Z_j, which have a rank of at most $2jr$ for $j = 1, \dots, s$. When the truncated SVD is used for retraction, the cost is dominated by the QR factorizations of the $n \times 2jr$ factors of Z_j. In summary, the total complexity for one time step of PRK is:

$$\mathcal{O}\left(\underbrace{s^3nr^2}_{\text{total retraction cost}} + \underbrace{snr_Fr}_{s\times\text{application of Pr}} + \underbrace{sc_F}_{s\times\text{evaluation of F}}\right)$$

When compared to the complexity of our method (4.9), we see that the term related to the projection, which is $\mathcal{O}(s^2nr_Fr)$ in our method, is reduced to $\mathcal{O}(snr_Fr)$. This difference becomes relevant when $r_F > sr$. The reason for this is that PRK can reuse computations for the tangent space projections across different stages. In contrast, our randomized low-rank RK method employs different sketches for each stage and thus cannot reuse these computations. As noted earlier, this issue can be addressed by reusing the random sketching matrices, though this approach currently lacks theoretical justification. However, since the number of stages s is typically very small, this issue may not be highly significant in practice. Our method utilizes sketching instead of tangent space projection, which offers two main advantages. As discussed earlier in this Section, the tangent space projection can introduce a significant error, even when the solution is well-approximated by a low-rank matrix. In such scenarios, our method is expected to be more accurate. This expectation is validated by the numerical experiments presented in Section 5. Since PRK method employs a fixed, deterministic tangent space projection to constrain the solution to the low-rank manifold so the RLRK endows the sketching approach with greater flexibility, as the random matrices can be adapted to exploit the specific structure of the problem at hand. This flexibility is particularly advantageous in following scenarios:

- **Block-Structured Matrices:** When the function F has a block structure, random sketches offer the adaptability needed to exploit this configuration. For instance, one can use a block SRFT to leverage parallel computing architectures, enhancing computational efficiency [**23**].
- **Nonlinear Structures:** Consider a scenario where $F(A)$ involves the Hadamard product of the matrix A with itself, a situation that can arise from quadratic or cubic nonlinearities in the underlying partial differential equation. Although the rank of the matrix stage F_j in the RLRK algorithm might be substantially larger, its factored representation often possesses a rich Kronecker product structure. This inherent structure can be effectively exploited when sketching F_j with Khatri-Rao products of random matrices [**27**; **32**].

4.3. **Error analysis.** With high probability, our method achieves a qualitative error behavior at least as good as that established for PRK. To see this, we follow [**1**; **3**] and consider the stage orders $\gamma_1, \ldots, \gamma_s$, which are defined as the local errors of the stages $\tilde{Z}_j$ in the standard RK method 2.3. For every $h \leq h_0$,

$$\|\tilde{Z}_j - \phi_F^{c_j h}(A_i)\|_F \leq C_L h^{\gamma_j + 1}, \quad j = 1, \ldots, s, \tag{4.12}$$

with $c_j = a_{j1} + a_{j2} + \cdots + a_{j,j-1}$. We then obtain the following result, which corresponds to Theorem 6 in [**3**].

Theorem 4.1. *Consider the randomized low-rank RK method 4.1 that utilizes independent standard Gaussian matrices, oversampling parameters $p, \ell \geq 4$, and an explicit s-stage RK method of order τ with stage orders $\gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_s$. Let*

$$\gamma = \begin{cases} \min(\tau, \gamma_2 + 1) & \text{if } b_2 \neq 0, \\ \min(\tau, \gamma_3 + 1, \gamma_2 + 2) & \text{if } b_2 = 0. \end{cases} \tag{4.13}$$

Then under the assumptions of Theorem 3.9, the global error is bounded for $q = \min\{p, \ell\}$ by

$$\|Y_N - A(Nh)\|_{L_q} \leq C(\delta + \epsilon + h^\gamma)$$

on the finite time interval $0 \leq Nh \leq T$, for all $h \leq h_0$. The constant C depends only on $L, T, h_0, s, C_L, C_N, \max_{i,j} |a_{ij}|$ and $\max_i |b_i|$. In particular, for any $\eta \geq 1$, it holds for fixed h and N that

$$\Pr\left[\|Y_N - A(Nh)\|_F \geq C\eta(\epsilon + h^\gamma + \delta)\right] \leq \frac{1}{\eta^q}.$$

Proof. The proof largely mirrors the arguments in [**3**]. The result is obtained by replacing the Frobenius norm with the L_q norm and introducing a few additional adjustments. A detailed proof is available in the appendix for the interested reader. □

Theorem 4.1 above shows that the randomized RK methods using the coefficients from the following RK methods of orders 1, 2, and 3 achieve the standard convergence order up to $\mathcal{O}(\epsilon)$:

- **RK 1 (Euler):** $b_1 = 1$,
- **RK 2 (Heun's method):** $a_{21} = 1, b_1 = b_2 = \frac{1}{2}$,

Unfortunately, for RK4, Theorem 4.1 only yields a convergence order of 2. Section 4.4 investigates this combination further.

Remark 4.2. As previously noted, the global nature of our three main assumptions (1.3, 1.4, and 1.5) can be quite limiting. If we modify these assumptions so that they only hold in a neighborhood of the exact solution $A(t)$ for $0 \leq t \leq T$, we must additionally ensure that Y_i and the intermediate stages remain within that neighborhood. Due to the presence of randomness, this complicates the analysis and yields slightly worse results. In what follows, we sketch the modifications required to localize these assumptions.

Suppose we want to ensure $E_i := \|Y_i - A(ih)\|_F \leq M$ for all $i = 1, \ldots, N-1$ to apply properties 1.3, 1.4, and 1.5 locally. (We simplify the discussion by ignoring the probability that the intermediate stages leave the neighborhood; it is straightforward to adapt the outlined approach by additionally requiring $\|Z_j - \tilde{Z}_j\|_F \leq M'$ for $j = 1, \ldots, s$, which would yield the same convergence rate with a different constant and a somewhat higher failure probability, increased by a factor of s.) To proceed, we can use the failure probability estimate from Theorem 4.1 to conclude

$$\Pr\left\{ E_i > M \,\middle|\, \bigcap_{j=0}^{i-1} \{E_j \leq M\} \right\} \leq \left(\frac{C(\epsilon + h^\gamma + \delta)}{M} \right)^q.$$

Using conditional probability and Bernoulli's inequality,

$$\begin{aligned}\Pr\left\{\bigcap_{j=0}^{N-1}\{E_j \le M\}\right\} &= \Pr\left\{E_{N-1} \le M \,\middle|\, \bigcap_{j=0}^{N-2}\{E_j \le M\}\right\}\Pr\left\{\bigcap_{j=0}^{N-2}\{E_j \le M\}\right\}\\ &\ge \left[1-\left(\frac{C(\epsilon+h^\gamma+\delta)}{M}\right)^q\right]\Pr\left\{\bigcap_{j=0}^{N-2}\{E_j \le M\}\right\}\\ &\ge \left[1-\left(\frac{C(\epsilon+h^\gamma+\delta)}{M}\right)^q\right]^{(N-1)}\\ &\ge 1-(N-1)\left(\frac{C(\epsilon+h^\gamma+\delta)}{M}\right)^q.\end{aligned}$$

Similarly, we have

$$\Pr\{\|Y_N - A(Nh)\|_F \ge C\eta(\epsilon+h^\gamma+\delta)\} \le \frac{1}{\eta^q} + (N-1)\left(\frac{C(\epsilon+h^\gamma+\delta)}{M}\right)^q.$$

The additional second term is small when M is large and/or $\epsilon+h^\gamma+\delta$ is small. To further quantify how this term affects the order of convergence, we substitute $\eta = \frac{M}{h^{1/q}}$ and assume $C(\epsilon+h^\gamma+\delta) \le 1$ and $h \le 1$, leading to

$$\Pr\left\{\|Y_N - A(Nh)\|_F \ge CM(\epsilon+h^\gamma+\delta)h^{-\frac{1}{q}}\right\} \le \frac{h}{M^q} + (N-1)\left(\frac{C(\epsilon+h^\gamma+\delta)}{M}\right)^q \le \frac{N}{M^q}. \tag{4.14}$$

The presence of the additional factor $h^{-\frac{1}{q}}$ reduces the convergence order by $\frac{1}{q}$. Since $q = \min\{p,\ell\}$, even modest choices for the oversampling parameters p and ℓ imply that this potential loss of order is negligible.

4.4. **Error Analysis for the Randomized Low-Rank RK4 Method.** While numerical experiments suggest that the randomized RK method based on RK4 achieves convergence order 4, a formal theoretical proof has been elusive. In this section, we establish order 4 convergence under the condition that the intermediate stages are oversampled. It is important to note that this oversampling is a theoretical device for the analysis and does not appear to be necessary in practice.

Specifically, we apply the coefficients of the classical RK4 method to the scheme in (14) and introduce oversampling for the intermediate stages

$$\begin{aligned}\hat{Z}_1 &= \mathcal{N}_1^{15r}(Y_i)\\ \hat{Z}_2 &= \mathcal{N}_2^{15r}\left(Y_i + \frac{h}{2}F(\hat{Z}_1)\right)\\ \hat{Z}_3 &= \mathcal{N}_3^{15r}\left(Y_i + \frac{h}{2}F(\hat{Z}_2)\right)\\ \hat{Z}_4 &= \mathcal{N}_4^{15r}(Y_i + hF(\hat{Z}_3))\\ Y_{i+1} &= \mathcal{N}_5^{r}\left(Y_i + \frac{h}{6}(F(\hat{Z}_1) + 2F(\hat{Z}_2) + 2F(\hat{Z}_3) + F(\hat{Z}_4))\right).\end{aligned} \tag{4.15}$$

In this formulation, $\mathcal{N}_5^r$ denotes the generalized Nyström method (8) with target rank r. In contrast, $\mathcal{N}_i^{15r}$ for $i = 1,\dots,4$ refers to the same method but with an increased target rank of $15r$. Since the intermediate stages may exit the manifold $\mathcal{M}_r$, a stronger low-rank approximability assumption is required. For $h \le h_0$, we assume:

$$\left\|\Phi_F^h(Y) - [[\Phi_F^h(Y)]]_k\right\|_F \le C_M h\epsilon, \quad \forall Y \in \mathcal{M}_k, \quad \forall r \le k \le 15r. \tag{4.16}$$

Our analysis of (4.15) begins with the following result, which establishes the low-rank approximability of F as a consequence of assumption (4.16).

Lemma 4.3. *Assuming that F satisfies (4.16), let k be any integer such that $r \le k \le 15r$. Then*

$$\|F(Y) - [[F(Y)]]_{2k}\|_F \le C_M\epsilon, \quad \forall Y \in \mathcal{M}_k.$$

Proof. By definition, $F(Y)$ is the time derivative of the flow $\Phi_t^F(Y)$. Therefore, for any $\gamma > 0$, there exists an $h > 0$ such that

$$\left\|\frac{\Phi_F^h(Y) - \Phi_F^0(Y)}{h} - F(Y)\right\|_F \le \gamma.$$

Since the term $[[\Phi_F^{h_j}(Y)]]_k - \Phi_F^0(Y)$ has a rank of at most $2k$, it follows that

$$\begin{aligned}\|F(Y) - [[F(Y)]]_{2k}\|_F &\leq \left\| F(Y) - \frac{[[\Phi_F \check{r} h_j(Y)]]_k - \Phi_F^0(Y)}{h_j} \right\|_F \\ &\leq \left\| F(Y) - \frac{\Phi_F^{h_j}(Y) - \Phi_F^0(Y)}{h_j} \right\|_F + \left\| \frac{\Phi_F^{h_j}(Y) - [[\Phi_F^{h_j}(Y)]]_k}{h_j} \right\|_F \\ &\leq \gamma + C_M \epsilon.\end{aligned}$$

The result is obtained by letting $\gamma \to 0$. □

The following auxiliary lemma provides a bound that will be instrumental in analyzing the local error of (4.15).

Lemma 4.4. *Let $Z \in \mathcal{M}_r, X \in \mathbb{R}^{m\times n}, \alpha \geq 0$, and $k \leq 14r$. Then*

$$\|Z + \alpha X - [[Z + \alpha X]]_{15r}\|_F \leq \alpha \|X - [[B]]_k\|_F$$

holds for any $B \in \mathbb{R}^{m\times n}$.

Proof. The result follows by noting that $Z + \alpha[[B]]_k$ has a rank of at most $15r$, which implies:

$$\|Z + \alpha X - [[Z + \alpha X]]_{15r}\|_F \leq \|Z + \alpha X - (Z + \alpha[[B]]_k)\|_F = \alpha \|X - [[B]]_k\|_F.$$

□

We can now establish a local error estimate for the oversampled scheme (4.15).

Lemma 4.5. *Suppose that the assumptions stated in Section 1 and (4.16) hold. Given $Y_i \in \mathcal{M}_r$, one step of the method (4.15) with independent standard Gaussian matrices and oversampling parameters $p, \ell \geq 4$ satisfies the local error estimate*

$$\|\Phi_F^h(Y_i) - Y_{i+1}\|_{L_q} \leq C(h\epsilon + h^5),$$

for $q = \min\{p, \ell\}$ and all $0 < h \leq h_0$. The constant C depends only on L, T, h_0, C_M, and C_N.

Proof. The proof involves bounding the moments of the differences between the stages $\hat{Z}_j$ of the oversampled method (4.15) and the stages $\tilde{Z}_j$ of the classical RK4 method applied to Y_i. The complete proof can be found in [**1**] □

Finally, the following theorem establishes the fourth-order convergence with respect to h for the modified randomized low-rank RK4 method (4.15). This result provides a theoretical justification for the convergence order observed in numerical experiments for the standard randomized low-rank RK4 method (i.e., without intermediate rank increases).

Theorem 4.6. *Suppose that the assumptions stated in Section 1 and (4.16) hold. The global error of the scheme (4.15) with independent standard Gaussian matrices and oversampling parameters $p, \ell \geq 4$ satisfies the bound*

$$\|Y_N - A(Nh)\|_{L_q} \leq C(\epsilon + h^4 + \delta),$$

for $q = \min\{p, \ell\}$, on the finite time-interval $0 \leq nh \leq T$ and for every $0 < h \leq h_0$. The constant C depends only on L, T, h_0, C_M and C_N. In particular, for any $\eta \geq 1$, it holds for fixed h and N that

$$\{\|Y_N - A(Nh)\|_F \geq C\eta(\epsilon + h^4 + \delta)\} \leq \frac{1}{\eta^q}.$$

Proof. From Lemma 4.5, we have

$$\left(\mathbb{E}\left(\mathbb{E}\left[\|\Phi_h^F(Y_i) - Y_{i+1}\|_F^q | Y_i\right]\right)\right)^{1/q} \leq \left(\mathbb{E}[C^q(h^5 + h\epsilon)^q]\right)^{1/q} = C(h^5 + h\epsilon).$$

Substituting this bound into (12) establishes the first part of the theorem. The second statement follows from an application of Markov's inequality. □

5. Numerical Experiments

In this chapter, we empirically validate the accuracy and efficiency of the randomized low-rank Runge-Kutta methods presented in Algorithm 4.1, which we will denote as Rand RK. We consider Rand RK1 (Euler) , Rand RK2, Rand RK4, and the Rand DOPRI45. The corresponding coefficients for each method are defined by their respective Butcher tableaus (see Section 2).

For the implementation of the rank reduction, we employ the generalized Nyström method as detailed in Section 3. The sketching process is typically performed using dense Gaussian random matrices, but for specific problems where structured sketches are advantageous, we utilize the Subsampled Randomized Fourier Transform (SRFT), which will be explicitly mentioned. The oversampling parameters are set to $p = \ell = \max\{2, 0.1r\}$ for Gaussian sketches and $p = \ell = \max\{10, 0.2r\}$ when using SRFT. The approximation error is measured by the Frobenius norm of the difference between the numerical solution and a high-precision reference solution. The reference is obtained by solving the full-dimensional matrix differential equation with MATLAB's solver ode45, using tight error tolerances of 'RelTol', 1e-12 and 'AbsTol', 1e-12.

Due to the stochastic nature of the Rand RK methods, we report the mean approximation error over 5 independent trials. The spread between the largest and smallest errors is indicated by error bars in the corresponding work-precision diagrams. For the initial two equation, we also compare our results against our implementation of the projected Runge-Kutta (PRK) method [**3**] for orders $s = 1, 2, 4$. The initial value for the PRK methods is obtained by applying the truncated SVD to the initial matrix A_0.

Although the generalized Nyström method is usually numerically stable [**20**], it may exhibit instabilities, especially when $\Psi^T Z\Omega$ is numerically rank-deficient. Such a phenomenon is observed in the third experiment involving the Lyapunov equation. In this case, we mitigate the instability by employing the ϵ-pseudoinverse, as defined in Equation (3.32), to regularize the computation.

All experiments in this section were performed in MATLAB (version R2025a). The code used to perform the experiments and produce the figures can be found at `https://github.com/`.

5.1. Lyapunov matrix differential equation (Heat equation).

Many matrix differential equations that serve as benchmarks for numerical methods originate from the spatial discretization of partial differential equations (PDEs) describing physical phenomena. In this section, we derive the Lyapunov matrix differential equation from its physical foundation in heat conduction, thereby motivating its use as a test problem. The physical derivation is based on the presentation in [**6**; **33**]. The description of diffusion processes, such as heat transfer, is founded on a physical conservation law. This principle states that the rate of change of a conserved quantity's density u (e.g., thermal energy) within a control volume D is equal to the net flux $\mathbf{J}$ across its boundary ∂D, plus any internal sources or sinks described by a density function f. In integral form, this is expressed as

$$\frac{d}{dt}\int_D \rho u \, d\mathbf{x} = -\int_{\partial D} \mathbf{J} \cdot d\mathbf{S} + \int_D \rho f \, d\mathbf{x}. \tag{5.1}$$

where ρ is the material density. By applying the Divergence Theorem, we can convert the surface integral into a volume integral, which leads to the differential form of the conservation law

$$\rho \frac{\partial u}{\partial t} = -\nabla \cdot \mathbf{J} + \rho f. \tag{5.2}$$

To form a closed system, a constitutive relation is required. **Fourier's Law of Heat Conduction** provides this by postulating that the heat flux is proportional to the negative gradient of the temperature, meaning heat flows from regions of higher temperature to lower temperature

$$\mathbf{J} = -\sigma \nabla u \tag{5.3}$$

Here, σ is the thermal conductivity. Substituting Fourier's Law into the conservation equation yields the general **heat equation**

$$\rho \frac{\partial u}{\partial t} = \nabla \cdot (\sigma \nabla u) + \rho f. \tag{5.4}$$

For a homogeneous and isotropic medium, σ is a constant scalar. The equation then simplifies to

$$\frac{\partial u}{\partial t} = \alpha \Delta u + g(\mathbf{x}, t). \tag{5.5}$$

where $\alpha = \sigma/\rho$ is the thermal diffusivity, $g = f$ is the source term, and $\Delta = \nabla \cdot \nabla$ is the Laplace operator.

5.1.1. *Discretization: From a PDE to a Matrix Differential Equation.* To solve the heat equation numerically, we discretize the continuous spatial domain into a grid of points. For simplicity, we consider the two-dimensional, non-homogeneous heat equation with $\alpha = 1$

$$\frac{\partial u}{\partial t} = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + g(x, y). \tag{5.6}$$

We introduce a uniform grid (x_i, y_j), and let $u_{ij}(t) \approx u(x_i, y_j, t)$ be the temperature at each grid point. The second-order spatial derivatives can be approximated using a central finite difference scheme:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{(x_i, y_j)} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} \tag{5.7}$$

This spatial discretization transforms the single PDE into a large system of coupled ordinary differential equations (ODEs), one for each grid point $u_{ij}(t)$. By arranging the unknown temperature values into a matrix $A(t) \in \mathbb{R}^{m\times n}$ where $(A)_{ij} = u_{ij}(t)$, the action of the discretized Laplacian can be represented by matrix multiplication. The system of ODEs can then be written in the compact matrix form

$$\dot{A}(t) = L_x A(t) + A(t) L_y^T + G. \tag{5.8}$$

Here, G is the matrix containing the values of the source term $g(x_i, y_j)$, and L_x and L_y are matrices representing the one-dimensional finite difference operators. For a uniform grid, they are symmetric tridiagonal matrices, e.g., $L_x = \frac{1}{(\Delta x)^2}\,\text{tridiag}(1, -2, 1)$.

In our case where the grid spacing is equal in all dimensions ($\Delta x = \Delta y$) and boundary conditions are symmetric, we have $L_x = L_y^T = L$, which simplifies the equation to

$$\dot{A}(t) = LA(t) + A(t)L^T + G. \tag{5.9}$$

It is important to note a simplification made in the formulation of the test problem as presented in the work of Lam, Ceruti, and Kressner [**1**]. While our physical derivation shows that the operator matrix L should be scaled by the factor $1/(\Delta x)^2$, the benchmark problem in the cited paper uses a simplified, unscaled version where $L = \text{tridiag}(1, -2, 1)$. In our numerical experiments, we will first adopt this simplified form of L to replicate the benchmark setup. Subsequently, we will conduct a second set of experiments using the physically-scaled operator $L = \frac{1}{(\Delta x)^2}\,\text{tridiag}(1, -2, 1)$ to analyze the performance of the methods on a problem where the matrix norms are dependent on the spatial discretization.

Having established the physical origins of the test problem, we now proceed to a direct numerical comparison between the Randomized Low-Rank (RLRK) and Projected (PRK) Runge-Kutta methods. For our first experiment, we consider the Lyapunov matrix differential equation as formulated in [**1**], which takes the form

$$\dot{A}(t) = LA(t) + A(t)L^T + \alpha\frac{C}{\|C\|_F}, \quad A(0) = A_0, \tag{5.10}$$

where $A(t) \in \mathbb{R}^{n\times n}$, $t \in [0, T]$, and we set $n = 128$. Consistent with the benchmark setup in [**1**], we use the symmetric, unscaled discrete Laplacian $L = \text{tridiag}(1, -2, 1)$. To define the source term C and the initial condition A_0, we follow a construction similar to the one used in [**6**]:

$$C_{ij} = \sum_{k=1}^{11} 10^{-(k-1)} \cdot e^{-k(x_i^2 + y_j^2)}, \tag{5.11}$$

$$(A_0)_{ij} = \sum_{k=1}^{20} b_k \cdot \sin(kx_i)\sin(ky_j), \quad \text{with } b_k = \begin{cases} 1, & \text{if } k = 1 \\ 5e^{-(7+0.5(k-2))}, & \text{if } k > 1, \end{cases} \tag{5.12}$$

where (x_i, y_j) are uniform discretization points of the square $[-\pi, \pi] \times [-\pi, \pi]$. The final time is set to $T = 1$ and the target rank for all low-rank integrators is fixed at $r = 10$.

We compare the RLRK and PRK methods for two distinct scenarios defined by the perturbation parameter $\alpha = 10^{-5}$ and $\alpha = 1$. The fundamental difference between the methods is the tangential projection error inherent to PRK, which is expected to be significantly larger for $\alpha = 1$. We therefore hypothesize that the performance of PRK methods will degrade in this case, while the RLRK methods, which do not rely on this deterministic projection, will remain robust. We fix the rank to $r = 10$ and compare the accuracy of Rand Euler, Rand RK2, and Rand RK4 with the corresponding projected methods, PRK1, PRK2, and PRK4. The solution matrix is of size $Y(t) \in \mathbb{R}^{128\times 128}$, and we use the unscaled discrete Laplacian $L = \text{tridiag}(1, -2, 1)$, consistent with the benchmark setup discussed previously. The initial condition Y_0 is a symmetric low-rank matrix with rapidly decaying singular values.

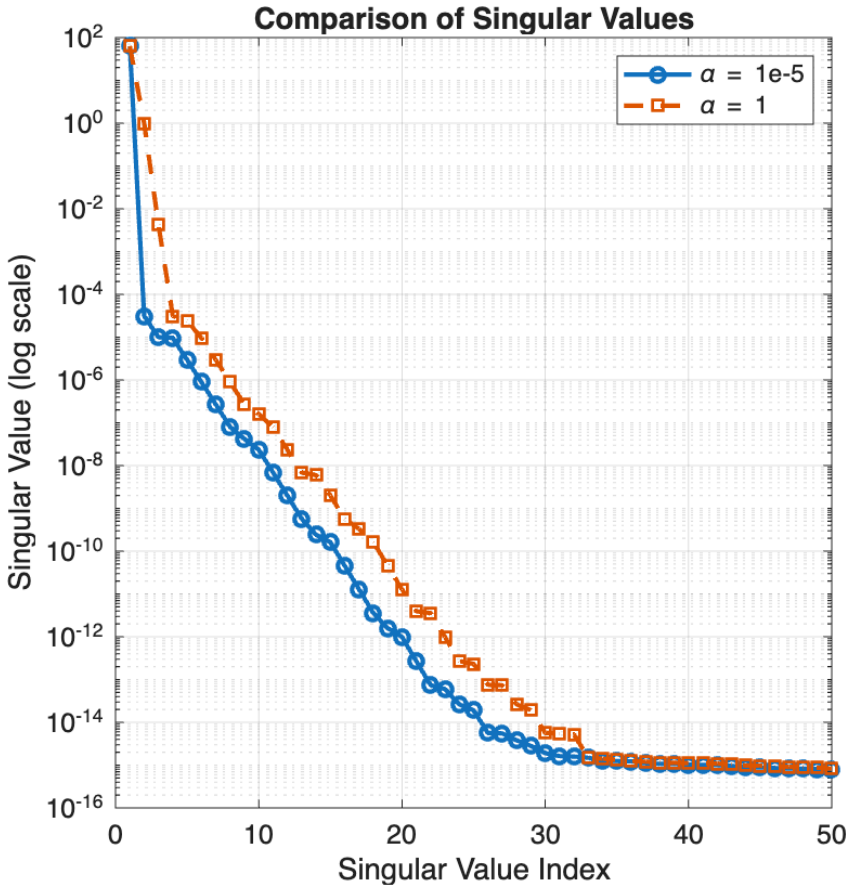

FIGURE 8. Singular values of the reference solution $Y(T)$ for $\alpha = 10^{-5}$ and $\alpha = 1$. The rapid decay in both cases justifies a low-rank approximation.

The accuracy of the PRK methods is fundamentally limited by the tangential projection error $\|F(Y_i) - \mathcal{P}_{T_{Y_i}\mathcal{M}_r}(F(Y_i))\|_F$, where Y_i is the approximation at the i-th time step. In Table 2, we report the average and maximum values of this error. For $\alpha = 1$, this error is several orders of magnitude larger than for $\alpha = 10^{-5}$. When the tangential projection error is large, the accuracy and convergence guarantees of PRK methods are lost.

This is indeed observed in the results presented in Figure 9. For $\alpha = 10^{-5}$, both PRK and Rand RK methods exhibit their expected first, second, and fourth orders of convergence, respectively, eventually saturating at the best approximation error floor. When $\alpha = 1$, however, all PRK methods fail to achieve their theoretical convergence rates and stagnate at a high error floor of approximately 10^{-4}, which is consistent with the large projection error shown in Table 2. On the other hand, the randomized methods remain robust. Rand Euler, Rand RK2, and Rand RK4 exhibit their respective first, second, and fourth-order convergence for both choices of α.

TABLE 2. Average and maximum tangential projection error for the Lyapunov matrix differential equation with $\alpha = 10^{-5}$ and $\alpha = 1$. The error is computed for the approximation Y_i at the i-th time step of PRK2 with $h = 5 \times 10^{-3}$.

α	10^{-5}	1
average $\|F(Y_i) - \mathcal{P}_{T_{Y_i}\mathcal{M}_r}(F(Y_i))\|_F$	1.3553×10^{-7}	4.0144×10^{-4}
max $\|F(Y_i) - \mathcal{P}_{T_{Y_i}\mathcal{M}_r}(F(Y_i))\|_F$	1.0×10^{-5}	0.7932

In conclusion, while PRK methods are highly efficient when the system dynamics stay close to the low-rank manifold, their accuracy is fundamentally limited by the projection error. The RLRK methods prove to be more robust, as their convergence behavior is not compromised by dynamics orthogonal to the manifold, making them a more reliable choice for problems where the low-rank structure of the dynamics is not guaranteed.

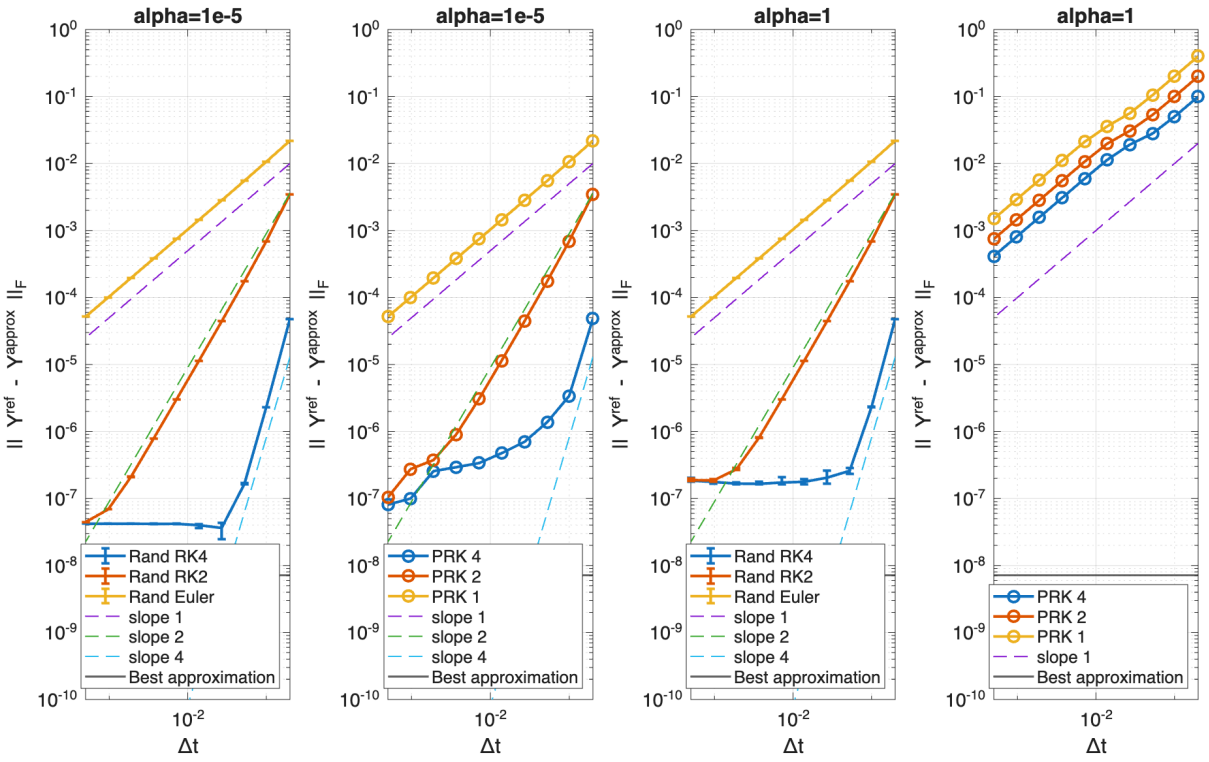

FIGURE 9. Error comparison between Randomized Runge-Kutta (RRK, first and third panels) and Projected Runge-Kutta (PRK, second and fourth panels) methods. The experiment is run for a weak perturbation ($\alpha = 10^{-5}$, left two panels) and a strong perturbation ($\alpha = 1$, right two panels). Dashed lines indicate theoretical convergence orders.

Having established the superior robustness of the Randomized Low-Rank (RLRK) methods over their projected counterparts, we now turn our attention to the efficiency and behavior of different time-stepping strategies within the RLRK framework itself. The following experiment directly compares a fixed-step integrator against an adaptive one to analyze the trade-offs in accuracy, computational work, and runtime. We consider the same Lyapunov matrix differential equation as in the previous section, with identical initial conditions and parameters. However, for this test, we set the target rank for this comparison to r=20 for both methods. The two solvers under investigation are a fixed-step, fourth-order randomized Runge-Kutta method (Rand RK4) and an adaptive randomized Dormand-Prince method (Rand Dopri45). The fixed-step method is executed with a range of step sizes, from $\Delta t \approx 2 \times 10^{-1}$ down to 5×10^{-4}, while the performance of the adaptive method is evaluated across a spectrum of user-defined error tolerances. The goal of this experiment is to create work-precision and time-precision diagrams to quantitatively assess which strategy offers a more efficient path to achieving a desired level of accuracy for this problem.

The results of this comparison are presented in Figure 10. The top two panels highlight the difference in stability and predictability between the methods. The top-left plot for the fixed-step **Rand RK4** shows that while the error initially follows the theoretical $O(\Delta t^4)$ convergence, it becomes highly erratic for step sizes smaller than 10^{-2}. Below this point, the error fluctuates unpredictably around a floor of 10^{-10}, making it impossible to reliably achieve a specific high accuracy by simply reducing the step size. In contrast, the top-right panel demonstrates the superior stability of the adaptive **Rand Dopri** method. The final error achieved corresponds almost perfectly to the user-specified tolerance, showcasing a controlled and predictable behavior across the entire accuracy range. This difference in stability directly translates to efficiency, as seen in the bottom two panels. The Work-Precision Diagram (bottom-left) shows that while both methods are comparable for moderate errors (down to $\approx 10^{-8}$), the adaptive method is far more efficient for achieving high accuracy. To reach an error below 10^{-9}, Rand RK4 requires a rapidly increasing number of function evaluations with no guarantee of success, whereas the adaptive method follows a smooth and efficient path. The Time-Precision Diagram (bottom-right) confirms this conclusion in terms of runtime. To achieve a final error of 10^{-10}, the adaptive method consistently takes

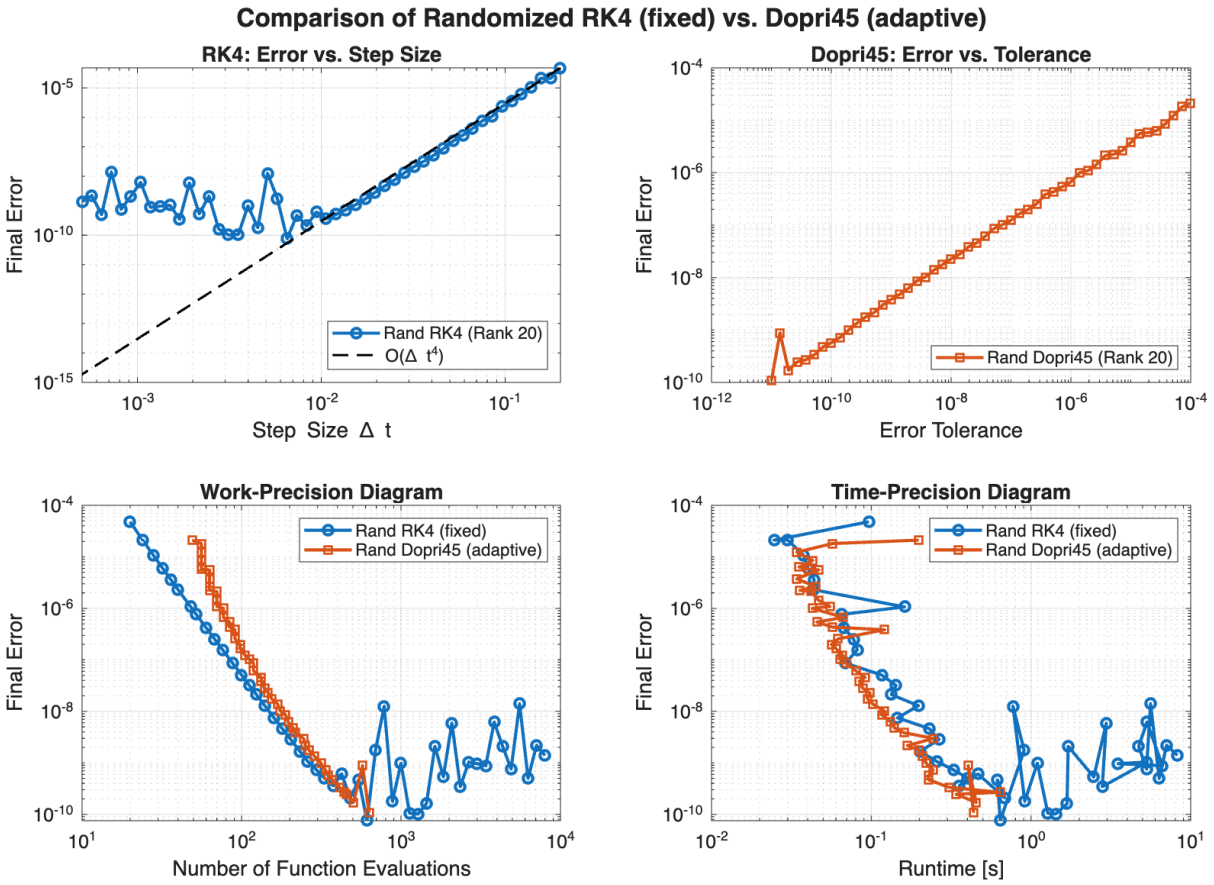

FIGURE 10. Comparison between the fixed-step Rand RK4 and the adaptive Rand Dopri45 method for the Lyapunov equation with a target rank of $r = 20$. The plots show error vs. step size/tolerance, as well as work-precision and time-precision diagrams.

around 1 second, while the runtime for the fixed-step method becomes highly unpredictable, potentially requiring up to 10 times longer. Therefore, the adaptive step-size strategy is not only more robust but is also the more computationally efficient choice when high accuracy is desired.

Now we consider again the same Lyapunov matrix differential equation as in the previous section, with identical initial conditions and parameters. However, for this test, we utilize the **physically-scaled** discrete Laplacian, $L = \frac{1}{(\Delta x)^2}\,\mathrm{tridiag}(1, -2, 1)$, to create a setting that more closely represents the underlying PDE. However, the introduction of the physically-scaled Laplacian, which contains the very small factor $(\Delta x)^2 \approx (2\pi/127)^2$ in its denominator, changes the numerical properties of the system, making it significantly stiffer. The initial attempt to solve this problem using the standard RLRK methods with moderate parameters resulted in a failure, as shown in Figure 11. While the singular values of the reference solution still decay rapidly, the convergence plots for Rand RK4, RK2, and Rand RK1 show that the integration is immediately unstable. The error explodes or stagnates at a very high level, and none of the methods achieve their theoretical convergence order.

To address this severe numerical instability, we attempted to stabilize the integration by significantly altering the solver parameters. The hypothesis was that the instability could be caused by an insufficient rank or by ill-conditioning in the Nyström step. We therefore implemented three changes simultaneously.

(1) The target rank was increased substantially to $r = 50$.
(2) The previously discussed ϵ-pseudoinverse was employed to regularize the Nyström method.
(3) The simulation was performed with extremely small step sizes and, for the adaptive method, very tight error tolerances.

The results of this modified approach are shown in Figure 12. At first glance, the outcome appears much better. Both the fixed-step Rand RK4 and the adaptive Rand Dopri45 now exhibit their expected convergence behavior. However, this success is confined to an extremely narrow and impractical range of very small step sizes (for RK4) and tolerances (for Dopri45). Paradoxically, for larger step sizes—where

the error decays. There is no clear mathematical justification for why the error is smaller for larger step sizes within this already tiny window. This suggests a fundamental incompatibility between the solver's current formulation and the stiff nature of the physically-scaled problem. Therefore, we conclude that while it is possible to force convergence in a very limited regime, the RLRK method in this form is not a sensible or robust tool for solving this particular Lyapunov equation.

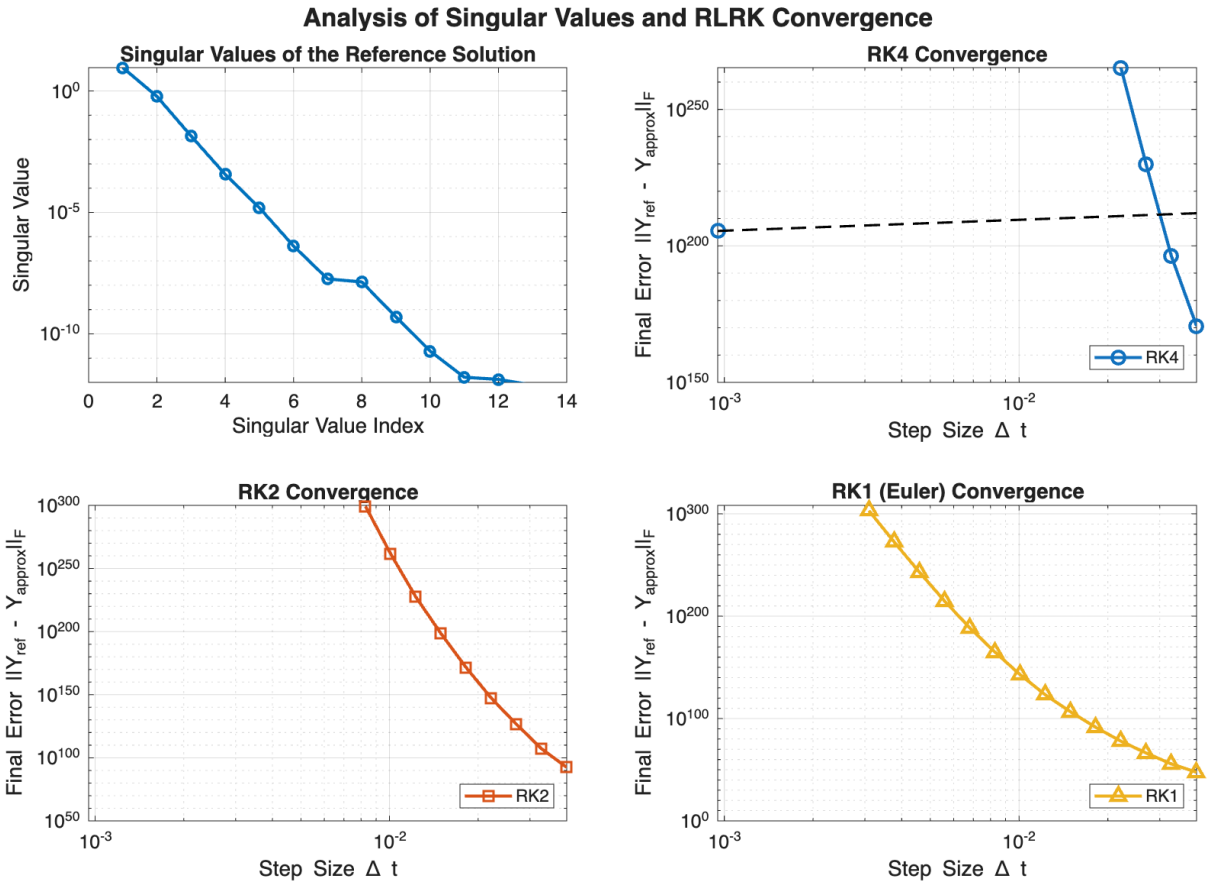

FIGURE 11. Initial failed attempt to solve the physically-scaled Lyapunov equation. All RLRK methods exhibit numerical instability.

5.1.2. *An Implementation Improvement for the Nyström Method.* Before presenting our next numerical results, we detail a implementation improvement over the baseline method presented in the work of Lam, Ceruti, and Kressner [**1**]. In their framework, the generalized Nyström method is used for rank reduction at each stage of the Runge-Kutta integrator. The standard implementation, reflected in the `matFull.m` function, reconstructs the full $r \times r$ rank-r matrix after computing its low-rank factors. To enhance efficiency, we implemented an optimized version, `matFull_svd.m`, which returns its constituent SVD factors (Q, U, S, V) packaged in a `struct`, avoiding the overhead of the full-size matrix. A comparison for the Lyapunov equation with $N = 128$ and $r = 40$ confirms that this optimization yields a significant speedup without altering the solution's accuracy. Despite this performance gain, for simplicity and to ensure direct comparability with the baseline approach, the following numerical experiments will continue to use the standard `matFull.m` implementation, as both methods are mathematically equivalent and produce identical results in terms of accuracy.

It is worth noting that for linear problems such as the Lyapunov equation, further optimization would be theoretically possible. One could reformulate the function F itself to operate directly on the factored representation of Y. This would eliminate the need for matrix reconstruction entirely, even within the function evaluation step, leading to an even greater reduction in computational cost.

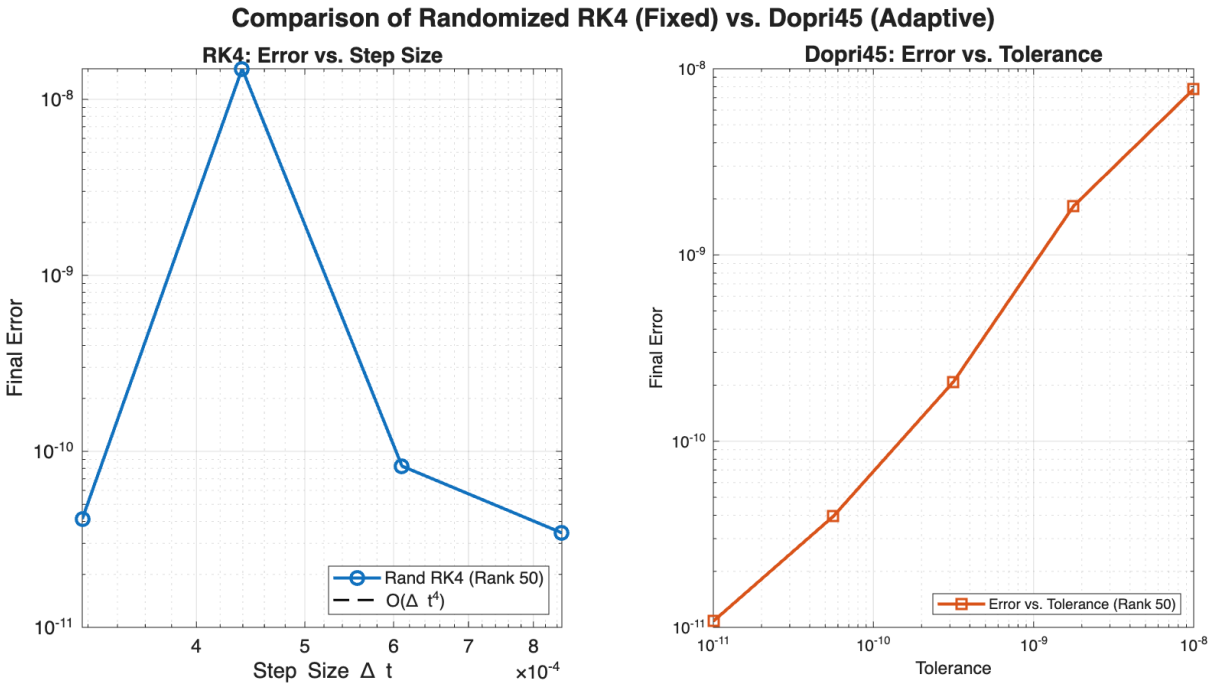

FIGURE 12. Results of the modified approach with a high rank ($r = 50$), ϵ-pseudoinverse, and extremely small step sizes/tolerances. Convergence is achieved, but only in a very narrow and impractical regime.

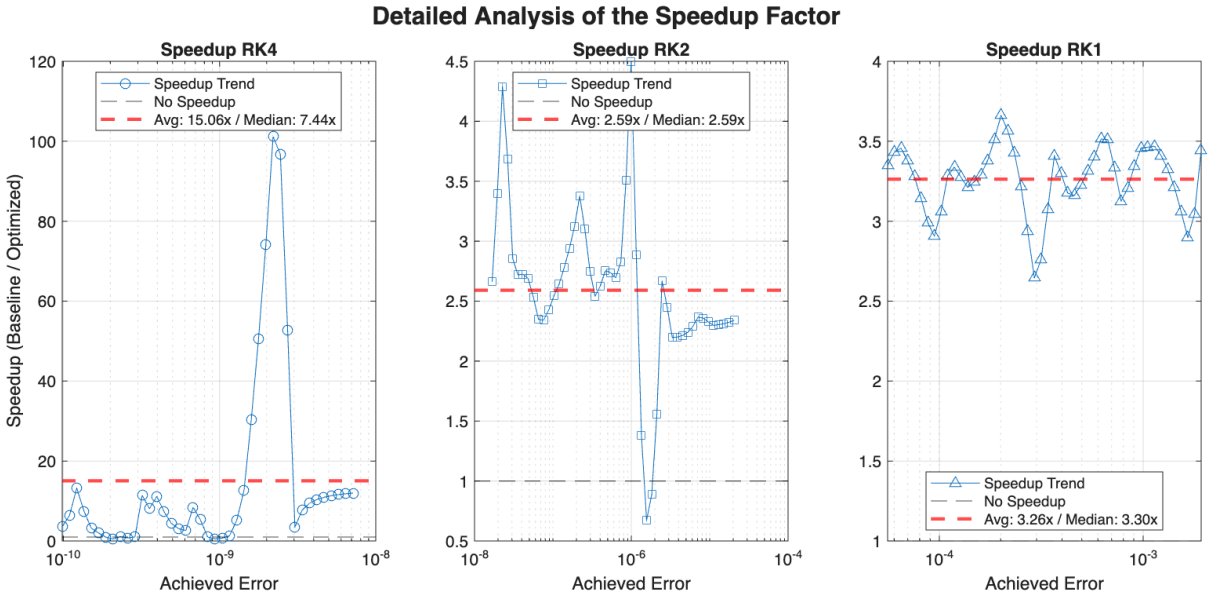

FIGURE 13. Speedup factor achieved by the optimized implementation compared to the baseline for RK4, RK2, and RK1 methods. The plot shows the ratio of baseline runtime to optimized runtime for a given achieved error.

5.2. **Non-linear Schrödinger equation.** We now consider the non-linear Schrödinger equation from [**2**; **Section 5.3**], where $A : [0, T] \to \mathbb{C}^{n\times n}$ evolves according to

$$\dot{A}(t) = \frac{i}{2}(BA + AB) + \alpha|A|^2 A. \tag{5.13}$$

The cubic nonlinearity $|A|^2A$ is taken element-wise and $B = \mathrm{diag}(1, 0, 1)$. We choose $n = 100$, $T = 5$, and the initial data

$$(A_0)_{ij} = \exp\left(-\frac{(i-60)^2}{100} - \frac{(j-50)^2}{100}\right) + \exp\left(-\frac{(i-50)^2}{100} - \frac{(j-40)^2}{100}\right).$$

In this example, we aim at computing approximations of rank up to 30. To ensure that A_0 has rank at least 30, we perturb the initial data by taking its full SVD and setting the singular values $3, 4, \ldots, 32$ to 10^{-9}.

First, we set $\alpha = 0.3$. The singular values of the reference solution at $t = T$, as well as the approximation errors by Rand Euler, Rand RK2, and Rand RK4 with different ranks, are shown in Figure 14. We again observe that Rand Euler achieves first-order convergence in time, Rand RK2 achieves second-order convergence, while Rand RK4 achieves fourth-order convergence until the error reaches the level of the low-rank approximation error. First, we test the performance of the randomized integrators where new, independent random matrices are drawn at each stage of the respective Runge-Kutta method. Figure 14 displays the results for Rand RK1 (with ranks 2, 4, 6, 8, 12), Rand RK2 (ranks 6, 10, 14, 18), and Rand RK4 (ranks 15, 20, 25, 30).

The convergence plots clearly demonstrate that the methods achieve their expected classical orders. Rand RK1, RK2, and RK4 exhibit first, second, and fourth-order convergence, respectively. This behavior persists until the temporal discretization error, which decreases with the time step Δt, becomes smaller than the spatial error floor. This error floor is determined by the chosen rank, and as expected, a higher rank results in a lower error plateau.

Next, we repeat the experiment with a some modifications. Within a single time step, the same set of sketching matrices is used for all internal stages of the RK methods ($\Omega_1 = \Omega_2 = \cdots = \Omega_{s+1}$ and $\Psi_1 = \Psi_2 = \cdots = \Psi_{s+1}$). The results of this approach are shown in Figure 15.

A comparison of the two figures reveals a substantial degradation in performance for the higher-order methods. While the first-order Rand RK1's convergence remains largely unaffected, Rand RK2 shows signs of instability, particularly for lower ranks where the error does not decrease monotonically with the time step. The effect is most pronounced for Rand RK4, which partially fails to achieve its theoretical fourth-order convergence. The error behaves erratically and does not follow the expected slope, indicating a breakdown of the method's accuracy.

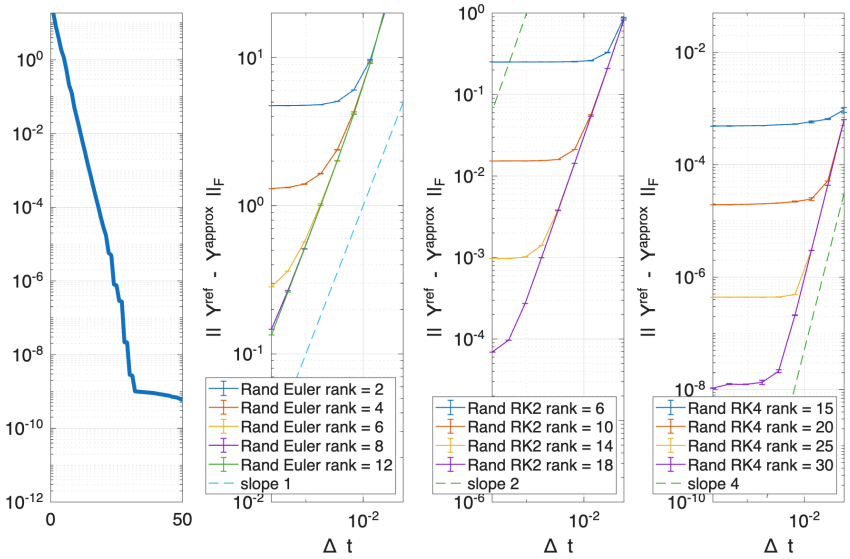

FIGURE 14. Performance of randomized integrators for the non-linear Schrödinger equation ($\alpha = 0.3$). The figure shows the singular value spectrum of the reference solution at $T = 5$, alongside a comparison of approximation errors. The errors are evaluated for the Rand RK1, Rand RK2, and Rand RK4 methods using various ranks and time-step sizes.

Finally, we repeat the experiment using Subsampled Randomized Fourier Transform (SRFT) matrices as the sketching operators (see Figure 16. This approach delivers excellent results, comparable to the original experiment where fresh random matrices were used in each stage. The accompanying figure clearly demonstrates that the methods achieve their expected convergence rates. The error plots for

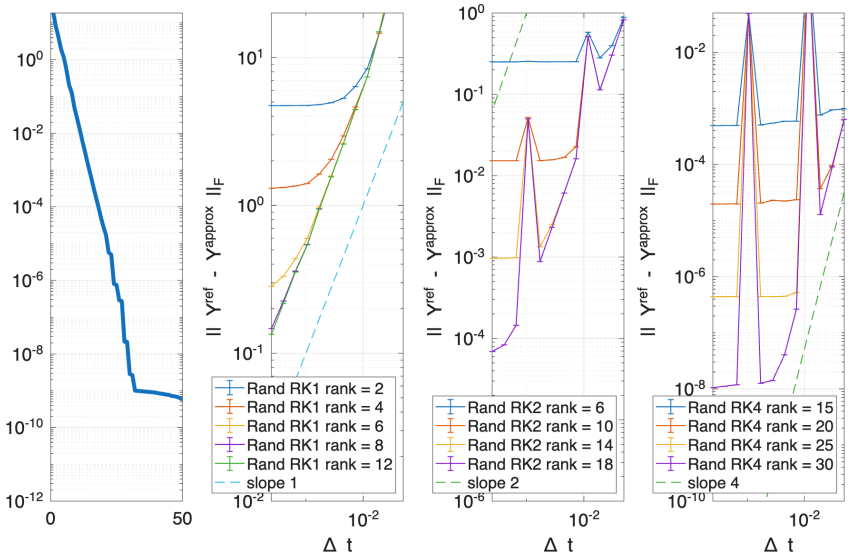

FIGURE 15. Performance of randomized integrators for the non-linear Schrödinger equation ($\alpha = 0.3$). The figure shows the singular value spectrum of the reference solution at $T = 5$, alongside a comparison of approximation errors. The errors are evaluated for the Rand RK1, Rand RK2, and Rand RK4 methods using various ranks and time-step sizes and constant matrices.

RLRK1, RLRK2, and RIRK4 precisely follow the reference slopes of 1, 2, and 4, respectively. For each method, the error decreases with the time step size until it stagnates at an error floor. This floor is dictated by the low-rank approximation error, and as expected, employing a higher rank leads to a more accurate result.

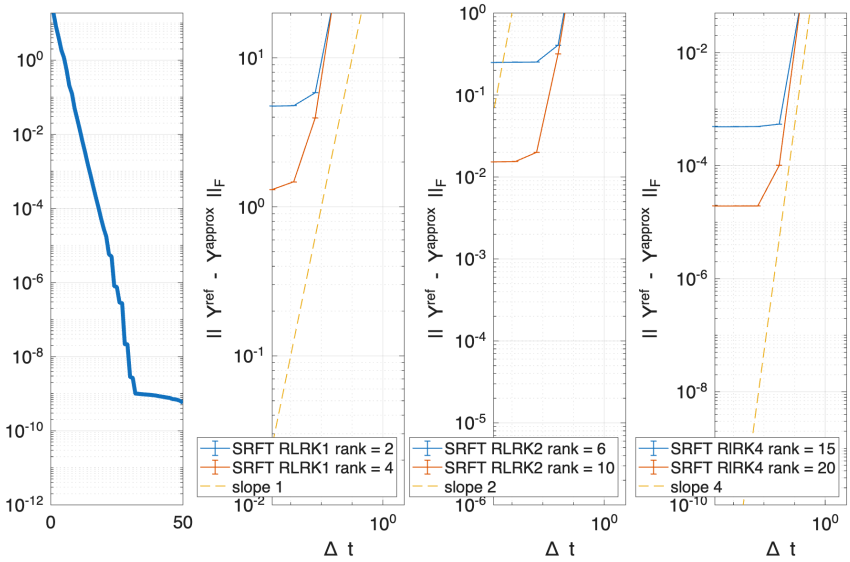

FIGURE 16. Performance of randomized integrators for the non-linear Schrödinger equation ($\alpha = 0.3$). The figure shows the singular value spectrum of the reference solution at $T = 5$, alongside a comparison of approximation errors. The errors are evaluated for the Rand RK1, Rand RK2, and Rand RK4 methods using various ranks and time-step sizes and SRFT matrices.

The following Figure 17 analyzes the impact of the oversampling parameter, p, on the performance of the Rand RK4 method for the nonlinear Schrödinger equation, with the rank fixed at $r = 30$. The left panel plots the final accuracy against the time step size for five different oversampling strategies ($p \in \{0, 1, 2, 4, 6\}$), while the right panel shows the corresponding total runtime for each strategy.

The accuracy plot on the left clearly demonstrates the necessity of oversampling. The strategies with no oversampling ($p = 0$) fail to achieve high accuracy, with errors stagnating above 10^{-6}. In contrast,

a improvement is observed for $p \geq 2$, where the method consistently reaches a much lower error floor, in the range of 10^{-8}. Further increasing the oversampling from $p = 2$ to $p = 4$ or $p = 6$ fails to provide a substantial gain in accuracy. Paradoxically, these higher values can even result in a slight increase in the final error, suggesting that excessive oversampling may introduce numerical noise without capturing additional essential dynamics of the system.

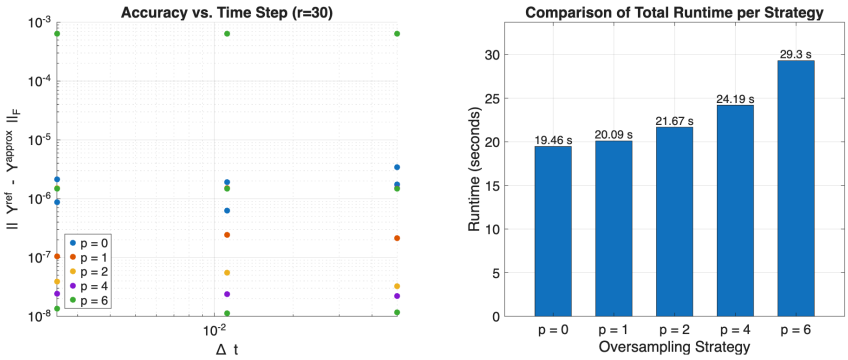

FIGURE 17. This figure analyzes the accuracy (left) and total runtime (right) of a Rand RK 4 with a fixed rank of r=30 used to solve the nonlinear Schrödinger equation. The performance is compared across five oversampling strategies (p=0,1,2,4,6).

Considering both accuracy and efficiency, an oversampling parameter of p=2 emerges as a sensible choice for this problem. It provides the leap in accuracy compared to lower values of p, while avoiding the significant additional runtime costs incurred by larger oversampling parameters like $p = 4$ or $p = 6$, which offer no substantial further improvement in the final error. Thus, $p = 2$ represents an effective trade-off between computational cost and accuracy.

5.3. **Discrete Schrödinger equation in imaginary time.** This example deals with the approximation of the solution to the discrete Schrödinger equation in imaginary time, following the work in [**2**]

$$\dot{A}(t) = -H[A(t)], \quad A(0) = A_0, \quad t \in [0, 0.5]. \tag{5.14}$$

where the Hamiltonian operator H is given by

$$H[A(t)] = -\frac{1}{2}(DA(t) + A(t)D) + V_{\cos}A(t)V_{\cos} \in \mathbb{R}^{n\times n}.$$

Here, $D = \text{diag}(-1, 2, -1)$ represents the discrete 1D Laplace operator, and $V_{\cos}$ is a diagonal matrix with entries $1 - \cos(2j\pi/n)$ for indices $j = -n/2, \ldots, n/2 - 1$. For the numerical simulation, we use a dimension of $n = 512$ and a random initial value A_0 whose singular values are prescribed to be 10^{-i} for $i = 1, \ldots, 512$. Figure 18 first plots the singular values of the reference solution at the final time $T = 0.5$. Subsequently, it shows a comparison of the approximation error for the Rand RK1, Rand RK2, and Rand RK4 methods using a fixed rank of 40. The top-right plot of 18 confirms that the fixed-step methods, Rand RK1, RK2, and RK4, achieve their respective theoretical first, second, and fourth-order convergence until the accuracy is limited by the rank-40 approximation error. The validity of this low-rank approach is supported by the rapidly decaying singular values of the reference solution shown in the top-left panel. The efficiency comparison in the bottom panels further reveals that while the fixed-step Rand RK4 integrator performs robustly for this problem, the adaptive Rand Dopri45 method fails to converge effectively, establishing the fixed-step solver as the more reliable choice in this specific scenario.

5.4. **Allen Cahn.** This section is inspired by the work of Benjamin Carrel and Bart Vandereycken , and A. Rodgers and D. Venturi [**3**]. It describes the process of phase separation in multicomponent alloy systems. In its simplest form, the partial differential equation is given by

$$\frac{\delta f}{\delta t} = \epsilon \Delta f + f - f^3.$$

where Δf is the diffusion term and $f - f^3$ is the reaction term. The stiffness is controlled by the small parameter ϵ. We consider the initial condition $f_0(x, y)$.

After discretization on a 2D grid, the solution $u(x, y, t)$ leads to a matrix $A(t)$. The Laplace operator is:

$$\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

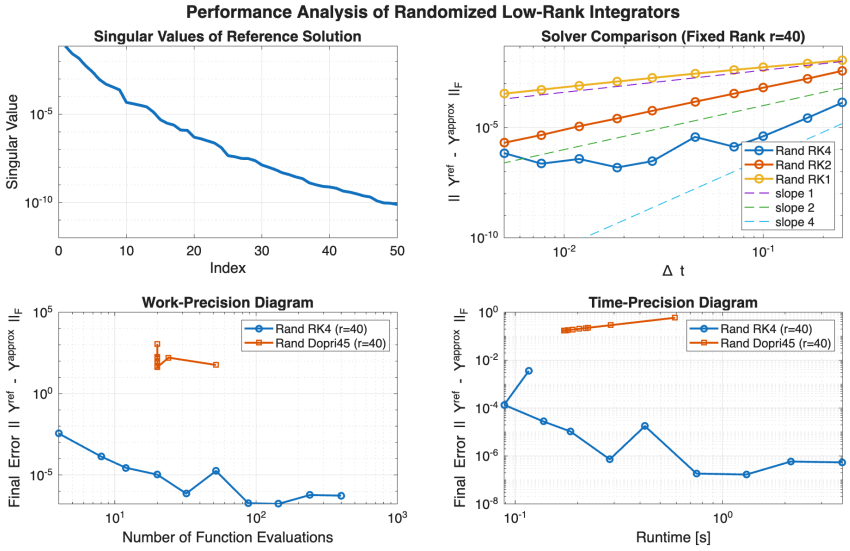

FIGURE 18. The figure displays (top-left) the singular values of the reference solution at T = 0.5 and (top-right) the approximation errors for the Rand Euler, Rand RK2, and Rand RK4 methods at a fixed rank of 40. The bottom panels compare the efficiency of a fixed-step integrator Rand RK4 versus an a(Rand Dopri45 via work-precision and time-precision diagrams.

The function $F(A)$ can be approximated up to rank r.

$$F(A) = \epsilon(D_{xx} + AD_{yy}) + \frac{1}{\epsilon}(A - A^3)$$

The matrix differential equation arising from discretizing the Allen-Cahn equation via finite differences is

$$\dot{A} = \epsilon(LA + AL) + A - A^3$$

with initial data:

$$(A_0)_{ij} = \frac{\left[e^{-\tanh^2(x_i)} + e^{-\tanh^2(y_j)}\right] \sin(\alpha \sin(\phi))}{1 + e^{i(x-s/2)\Gamma} + e^{i(x-s/2)|}}$$

Here, A^3 is the element-wise product, and $(x_i, y_j) \in [0, 2\pi]^2$ are uniform discretization points.

We compare the standard fixed-step fourth-order Runge-Kutta (RK4) method against the adaptive Dormand-Prince (Rand DOPRI45) method. The objective is to observe the stability and accuracy of these methods under such conditions. The figure 19 displays a time-series comparison of a high-fidelity reference solution against the solutions generated by the RK4 and DOPRI45 methods. The top-to-bottom rows show the state of the system at times $t = 1, 3, 5, 7$, and 10. The columns show the reference solution, the results from RK4 and DOPRI45, and their respective differences (errors) relative to the reference. The second column shows that the RK4 integrator successfully reproduces the dynamics of the phase separation, closely matching the reference solution at all time steps. The fourth column quantifies this observation, showing a very small relative error throughout the simulation. This indicates that the RK4 method is robust and accurate for this problem, even with a very small step size. In contrast, the third column (Rand DOPRI45) reveals a complete failure of the DOPRI45 method. After the first time step, the solution degenerates into numerical noise and fails to capture any of the physical dynamics. The fifth column confirms this, showing a large and uniformly distributed relative error, signifying that the computed solution has no correlation with the true solution.

This experiment demonstrates a critical difference in the behavior of the two solvers under the chosen conditions. The fixed-step RK4 method remains stable and reliable. However, the adaptive Rand DOPRI45 method, which is designed to adjust its step size to meet a certain error tolerance, appears to become unstable when forced to operate with an extremely small, fixed step size. This can lead to the accumulation of floating-point errors and numerical divergence, as seen in the plot. The result underscores the importance of not only choosing an appropriate integration scheme but also ensuring its parameters are suited to the specific problem and computational constraints.

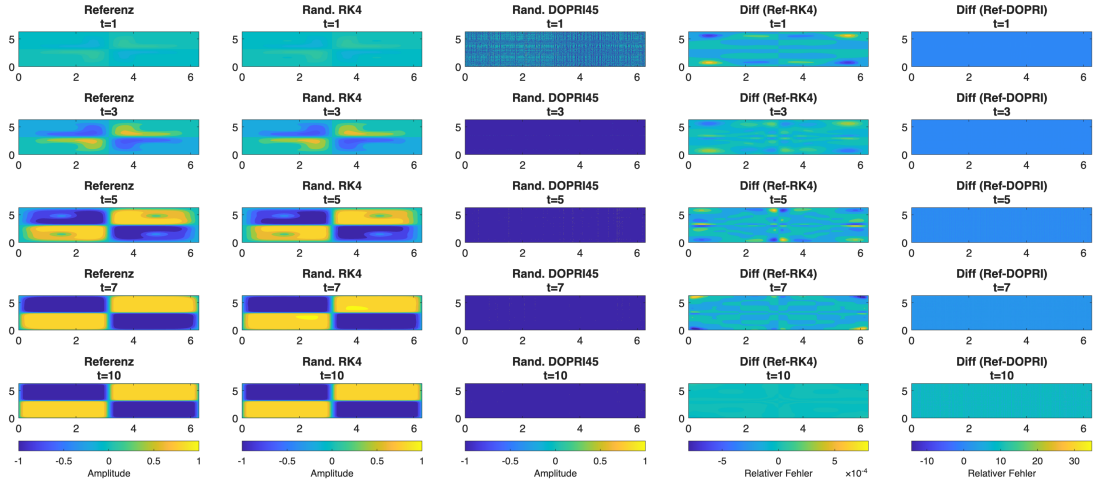

FIGURE 19. Performance of randomized integrators for the allen cahn equation . The figure shows reference solution at $T = 10$, alongside a comparison of approximation errors of Rand RK4 and Rand Dopri45.

CONCLUSION

This thesis addressed the computational challenges posed by high-dimensional matrix differential equations, where standard numerical integrators are limited by prohibitive rank growth. It conducted a thorough analysis of Randomized Low-Rank Runge-Kutta (RLRK) methods, an approach that circumvents this limitation by embedding randomized projections directly into the time-stepping scheme. The core contributions included a theoretical error analysis, the implementation and comparison of sketching techniques using both dense Gaussian matrices and the more efficient Subsampled Randomized Fourier Transform (SRFT), and a critical evaluation of fixed-step versus adaptive solvers. The numerical experiments confirmed the superior robustness of RLRK methods, particularly in scenarios where Projected Runge-Kutta (PRK) methods falter due to large tangential projection errors, as demonstrated with the Lyapunov equation. Furthermore, the results revealed a strong problem-dependency regarding time-stepping strategies. While the adaptive Rand Dopri45 method showed higher efficiency for some problems, fixed-step integrators proved significantly more stable for stiff systems like the Allen-Cahn equation.

A theoretical insight reinforced by this work is the advantage of relying on the low-rank approximability of the solution flow rather than the restrictive tangentiality condition of the vector field required by PRK methods. Example 3.6 provided a demonstration of this, showing a scenario where the tangent space projection error was several orders of magnitude larger than the actual low-rank approximation error of the flow, thereby explaining the fundamental limitation of the PRK approach. Methodologically, the choice of the generalized Nyström method for rank reduction proved to be useful. Unlike the more common Randomized SVD, its linearity allows for operating on small sketch matrices directly, avoiding the computationally prohibitive step of forming large, high-rank intermediate matrices and thus substantially reducing both memory and computational costs. This choice was a cornerstone of the method's efficiency.

Beyond the theoretical framework, this thesis yielded several concrete insights for the practical implementation of RLRK methods. The experiments with the non-linear Schrödinger equation demonstrated that the strategy of reusing the same random sketching matrices within a single time step, while tempting for performance reasons, leads to a significant degradation of accuracy and stability for higher-order methods like RK4. This underscores the importance of stochastic independence between the stage projections. Furthermore, the investigation of oversampling parameters showed that while some level of oversampling is essential for accuracy (e.g., $p \geq 2$), excessive oversampling does not provide further benefits and unnecessarily increases computational cost, establishing a clear trade-off for practitioners. The successful application of SRFT matrices also confirmed their role as a highly efficient, scalable alternative to dense Gaussian matrices, especially for very large problem dimensions.

Despite these successes, this work also identified several limitations and avenues for future research. While our numerical results strongly suggest that methods like Rand RK4 achieve their classical convergence orders, a more direct theoretical proof is needed, as the current analysis required an artificial increase in the intermediate ranks not used in practice. Similarly, a formal proof of convergence for the

adaptive Rand Dopri45 method remains an open area for research. The experiments with the physically-scaled Lyapunov equation, and similarly with the Allen-Cahn equation, also revealed the limitations of the employed explicit solvers-particularly the adaptive one—when faced with severe stiffness. This often led to numerical instability and suggests a critical need for the development of implicit or more stability focused RLRK integrators to handle such problems robustly.

Future investigations should also address practical implementation questions and explore broader applications. For instance, the strategy of using constant sketching matrices ($\Omega_1 = \Omega_2 = \cdots = \Omega_{s+1}$) within a time step, while computationally attractive, can lead to a degradation in accuracy for higher-order methods due to the loss of stochastic independence. While initial experiments on oversampling for Gaussian matrices provided a sensible choice for the investigated problem, further research is needed to develop robust heuristics for determining optimal parameters, particularly for SRFT matrices and across a wider range of applications. A significant practical challenge that remains is the a priori selection of the target rank r, which is often unknown. Developing adaptive strategies to dynamically adjust the rank during integration would greatly enhance the usability of these methods. Moreover, the potential of other random matrices should be explored. This includes block-structured matrices, like the block SRFT, which can leverage parallel computing architectures when the function F has a corresponding block structure. For problems with nonlinear structures, such as those arising from quadratic terms in PDEs that lead to Hadamard products, the underlying Kronecker product structure of the matrix stages could be exploited by sketching with Khatri-Rao products of random matrices. Addressing these challenges will be crucial for broadening the applicability of RLRK methods and solidifying their role as standard tools in scientific computing.

Appendix A. Appendix

A.1. **Proofs of technical results.**

Theorem A.1. *(Theorem 11) Let $\Omega \in \mathbb{R}^{n\times(r+p)}$ be a Gaussian random matrix and B be a fixed matrix of compatible size. Suppose $E(t) \in C([t_0, t_0+T], \mathbb{R}^{m\times n})$ such that $E(t_0) = 0$ and $E(t)$ is Lipschitz continuous with a Lipschitz constant $L \geq 0$ in the Frobenius norm. Then for any $u \geq 3$,*

$$Pr\left\{\sup_{t\in[t_0,t_0+T]} \|E(t)\Omega B\|_F > 2LT\|B\|_F(1+4u)\right\} \leq e^{-u^2/2}.$$

Proof. The proof is based on the **chaining method** to control the probability of the supremum of a stochastic process.

Let $X(t) = E(t)\Omega B$ for $t \in [t_0, t_0+T]$. For any two fixed time points $s, t \in [t_0, t_0+T]$, we define a function $h(\Omega) := \|X(t)-X(s)\|_F = \|(E(t)-E(s))\Omega B\|_F$. To use the concentration property of Gaussian matrices, we determine the Lipschitz constant of $h(\Omega)$.

The Lipschitz constant L_h for the function $h(\Omega)$ is determined as follows:

$$\begin{aligned}|h(\Omega_1) - h(\Omega_2)| &= |\|(E(t)-E(s))\Omega_1 B\|_F - \|(E(t)-E(s))\Omega_2 B\|_F| \\ &\leq \|(E(t)-E(s))(\Omega_1-\Omega_2)B\|_F \quad \text{(by the reverse triangle inequality)} \\ &\leq \|E(t)-E(s)\|_F\|\Omega_1-\Omega_2\|_F\|B\|_F \quad \text{(by sub-multiplicativity)}.\end{aligned}$$

This gives the Lipschitz constant:

$$L_h = \|E(t)-E(s)\|_F\|B\|_F. \tag{A.1}$$

According to Proposition 10.3 from Halko, Martinsson, and Tropp [**5**], for a Gaussian random matrix Ω and a Lipschitz continuous function h, the following inequality holds:

$$\Pr\{h(\Omega) \geq \mathbb{E}[h(\Omega)] + L_h \cdot u\} \leq e^{-u^2/2}.$$

Now, we must estimate the expectation $\mathbb{E}[h(\Omega)] = \mathbb{E}[\|(E(t)-E(s))\Omega B\|_F]$. Using Jensen's inequality, we have $\mathbb{E}[Y] \leq \sqrt{\mathbb{E}[Y^2]}$. This implies:

$$\mathbb{E}[\|(E(t)-E(s))\Omega B\|_F] \leq \sqrt{\mathbb{E}[\|(E(t)-E(s))\Omega B\|_F^2]}.$$

For a Gaussian random matrix Ω and fixed matrices A, B, the property $\mathbb{E}[\|A\Omega B\|_F^2] = \|A\|_F^2\|B\|_F^2$ holds. Therefore

$$\mathbb{E}[\|(E(t)-E(s))\Omega B\|_F] \leq \sqrt{\|E(t)-E(s)\|_F^2\|B\|_F^2} = \|E(t)-E(s)\|_F\|B\|_F.$$

Substituting this into the probability inequality, we get:

$$\mathbb{E}[h(\Omega)] + u\cdot L_h \leq \|E(t)-E(s)\|_F\|B\|_F + u\cdot(\|E(t)-E(s)\|_F\|B\|_F) = (1+u)\|E(t)-E(s)\|_F\|B\|_F.$$

Since $E(t)$ is Lipschitz continuous with constant L, $\|E(t)-E(s)\|_F \leq L|t-s|$. Thus

$$\mathbb{E}[h(\Omega)] + u\cdot L_h \leq (1+u)L|t-s|\|B\|_F.$$

This gives us the increment probability

$$\Pr\{\|X(t)-X(s)\|_F \geq (1+u)L|t-s|\|B\|_F\} \leq e^{-u^2/2}.$$

We use the chaining method. We consider a sequence of partitions T_n of the interval $[t_0, t_0+T]$. For $n=0$, we have $T_0 = \{t_0\}$, and for $n \geq 1$, T_n has 2^{2^n} points. Let $\pi_n(t)$ be the closest point to t in the partition T_n. We can write the increment $X(t)-X(t_0)$ as a telescoping series

$$X(t)-X(t_0) = \sum_{n=1}^{\infty}(X(\pi_n(t)) - X(\pi_{n-1}(t))).$$

By the triangle inequality, it holds that

$$\|X(t)-X(t_0)\|_F \leq \sum_{n=1}^{\infty}\|X(\pi_n(t)) - X(\pi_{n-1}(t))\|_F.$$

The distance between two consecutive points in the partition T_n is at most $\frac{T}{2^{2^{n-1}}}$. Since $E(t)$ is Lipschitz continuous, we have

$$\|E(\pi_n(t)) - E(\pi_{n-1}(t))\|_F \leq L|\pi_n(t)-\pi_{n-1}(t)| \leq L\frac{T}{2^{2^{n-1}}}.$$

□

References

[1] Hei Yin Lam, Gianluca Ceruti, and Daniel Kressner. *Randomized low-rank Runge-Kutta methods*. September 11, 2024.

[2] Gianluca Ceruti, Lukas Einkemmer, Jonas Kusch, and Christian Lubich. *A robust second-order low-rank integrator*. Preprint, 2024.

[3] Emil Kieri and Bart Vandereycken. *Projection methods for dynamical low-rank approximation of high-dimensional problems*. Comput. Methods Appl. Math., 19(1):73–92, 2019.

[4] Jonas Kusch. *Second-order robust parallel integrators for dynamical low-rank approximation*. arXiv preprint arXiv:2403.02834, 2024.

[5] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[6] Markus Bachmayr. *Numerische Analysis III. Skriptum zur Vorlesung im Wintersemester 2024/25*. 22. Januar 2025.

[7] Martin Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Vieweg und Teubner, 2008.

[8] Wolfgang Walter. *Gewöhnliche Differentialgleichungen*. Springer, 2000.

[9] Maria G. Westdickenberg. *Gewöhnliche Differentialgleichungen, SoSe 2024*. Lecture notes, RWTH Aachen University, 2024.

[10] J. R. Dormand and P. J. Prince. *A family of embedded Runge–Kutta formulae*. Journal of Computational and Applied Mathematics, 6(1):19–26, 1980.

[11] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Second edition, Springer Series in Computational Mathematics, Vol. 8, Springer, 1993.

[12] Jonas Kusch. *Second-order robust parallel integrators for dynamical low-rank approximation*. arXiv preprint arXiv:2403.02834, 2024.

[13] Gianluca Ceruti, Lukas Einkemmer, Jonas Kusch, and Christian Lubich. *A robust second-order low-rank BUG integrator based on the midpoint rule*. BIT, 64(3):Paper No. 30, 2024.

[14] D. Appelö and Y. Cheng. *Robust implicit Adaptive Low Rank Time-Stepping Methods for Matrix Differential Equations*. arXiv preprint arXiv:2402.05347, 2024.

[15] H. Babaee, M. Choi, T. P. Sapsis, and G. E. Karniadakis. *A robust bi-orthogonal/dynamically-orthogonal method using the covariance pseudo-inverse with application to stochastic flow problems*. J. Comput. Phys., **344**:303–319, 2017.

[16] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*. Volume 31 of Springer Series in Computational Mathematics. Springer, Heidelberg, 2010.

[17] A. Smilde, R. Bro, and P. Geladi. *Multi-Way Analysis: Applications in the Chemical Sciences*. Wiley, West Sussex, England, 2004.

[18] S. Schotthöfer, E. Zangrando, J. Kusch, G. Ceruti, and F. Tudisco. *Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations*. Advances in Neural Information Processing Systems, **35**:20051–20063, 2022.

[19] Yuji Nakatsukasa. *Fast and stable randomized low-rank matrix approximation*. arXiv preprint arXiv:2009.12238, September 25, 2020.

[20] E. Kieri, C. Lubich, and H. Walach. *Discretized dynamical low-rank approximation in the presence of small singular values*. SIAM J. Numer. Anal., **54**(2):1020–1038, 201

[21] A. Trombettoni and A. Smerzi. *Discrete solitons and breathers with dilute bose-einstein condensates*. Physical Review Letters, **86**(11):2353, 2001.

[22] C. F. Van Loan. The ubiquitous Kronecker product. *J. Comput. Appl. Math.*, 123 (2000), pp. 85–100.

[23] O. Balabanov, M. Beaupère, L. Grigori, and V. Lederer. *Block subsampled randomized Hadamard transform for Nyström approximation on distributed architectures*. In Proceedings of the 40th International Conference on Machine Learning, 2023.

[24] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), no. 90, 297–301.

[25] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM REVIEW*, 51(3) (2009), pp. 455–500.

[26] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 4th edition, 2013.

[27] Z. Bujanović, L. Grubišić, D. Kressner, and H. Y. Lam. *Subspace embedding with random Khatri-Rao products and its application to eigensolvers*. arXiv preprint arXiv:2405.11962, 2024.

[28] T. Atlagh, *RLRK: Implementation of Randomized Low-Rank Runge-Kutta Methods*, GitHub repository, 2025, https://github.com/Taha1703/RLRK.

[29] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin, Heidelberg, 2nd rev. edition, 1993.

[30] P. Kramer and M. Saraceno. *Geometry of the Time-Dependent Variational Principle in Quantum Mechanics*, volume 140 of Lecture Notes in Physics. Springer, Berlin, 1981.

[31] P. A. M. Dirac. *Note on exchange phenomena in the Thomas atom*. Mathematical Proceedings of the Cambridge Philosophical Society, **26**:376–385, 1930.

[32] D. Kressner and L. Periša. *Recompression of Hadamard products of tensors in Tucker format*. SIAM J. Sci. Comput., **39**:A1879–A1902, 2017. Klar, hier ist der Eintrag als bibitem für deine LaTeX-Datei.

[33] L. C. Evans, *Partial Differential Equations*, Graduate Studies in Mathematics, vol. 19 (2nd ed.), American Mathematical Society, Providence, RI, 2010.

[34] M. Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Vieweg und Teubner, 2008.