mathematics

IMPACT FACTOR 2.3

CITESCORE 4.0

Article

Koopman Spectral Linearization vs. Carleman Linearization: A Computational Comparison Study

Dongwei Shi and Xiu Yang

Special Issue
Quantum Computing and Scientific Computing

Edited by
Dr. Xiu Yang

MDPI

https://doi.org/10.3390/math12142156

mathematics

Article

Koopman Spectral Linearization vs. Carleman Linearization: A Computational Comparison Study

Dongwei Shi and Xiu Yang *

Department of Industrial and System Engineering, Lehigh University, Bethlehem, PA 18015, USA; dos222@lehigh.edu
* Correspondence: xiy518@lehigh.edu

Abstract: Nonlinearity presents a significant challenge in developing quantum algorithms involving differential equations, prompting the exploration of various linearization techniques, including the well-known Carleman Linearization. Instead, this paper introduces the Koopman Spectral Linearization method tailored for nonlinear autonomous ordinary differential equations. This innovative linearization approach harnesses the interpolation methods and the Koopman Operator Theory to yield a lifted linear system. It promises to serve as an alternative approach that can be employed in scenarios where Carleman Linearization is traditionally applied. Numerical experiments demonstrate the effectiveness of this linearization approach for several commonly used nonlinear ordinary differential equations.

Keywords: quantum algorithms; differential equations; Koopman operator; Carleman linearization

MSC: 65N35; 34B15; 35Q99; 68Q12

Citation: Shi, D.; Yang, X. Koopman Spectral Linearization vs. Carleman Linearization: A Computational Comparison Study. *Mathematics* **2024**, *12*, 2156. https://doi.org/10.3390/math12142156

Academic Editor: Davide Valenti

Received: 31 May 2024
Revised: 6 July 2024
Accepted: 6 July 2024
Published: 9 July 2024

1. Introduction

Differential equations find extensive applications in diverse fields such as fluid dynamics, biology, and finance. For instance, in fluid dynamics, the Navier–Stokes equations serve as fundamental tools for modeling fluid behavior [1]. In epidemiology, ordinary differential equations, such as those based on the Susceptible–Infectious–Recovered (SIR) model, are indispensable for analyzing the spread of infectious diseases [2]. Additionally, in finance, the Black–Scholes model relies on differential equations to facilitate option pricing [3]. However, computing solutions for high-dimensional differential equations presents a significant challenge. This challenge arises from the curse of dimensionality, wherein traditional numerical methods entail discretizing equations at grid points. As the dimension of equations increases, the number of grid points grows exponentially. Consequently, even basic linear algebra computations within these numerical algorithms become prohibitively complex.

A promising strategy for mitigating this complexity involves using quantum computing techniques since quantum computer leverages unique features such as quantum entanglement and superposition, enabling exponential speedups in specific computational tasks. As previous research has pointed out, replacing the linear systems solvers in classical numerical methods with their quantum counterparts, the Quantum Linear Systems Algorithm (QLSA), [4] can yield provably efficient quantum algorithms for solving linear ordinary differential equations (ODEs) [5–8] and linear Partial Differential Equations (PDEs) [7,9–11]. Additionally, quantum algorithms based on Hamiltonian simulation, rather than the QLSA, have also been theoretically shown to efficiently solve computational problems involving linear differential equations, whether for the standard Hamiltonian simulation or non-Hermitian Hamiltonian simulation [12].

However, despite a series of theoretical successes in developing quantum algorithms for linear differential equations, addressing nonlinear cases remains an open problem.

Early attempts [13] to directly combine QLSA with nonlinear ODEs proved to have poor scaling with evolution time. Recently, Liu et al. [14] demonstrated significant improvements by introducing Carleman Linearization to certain nonlinear ODEs. Using QLSA on the linearized differential equations greatly enhanced the complexity bound with respect to the evolution time. This approach has motivated many researchers to consider first employing linearization techniques, such as Carleman Linearization [15–17], Koopman von Neumann linearization [18–20] and the level set method [21], to approximate or represent low-dimensional nonlinear differential equations with high-dimensional linear ones, and then using quantum algorithms designed for linear differential equations to solve these problems. Figure 1 illustrates the work flow of these various quantum algorithms for solving nonlinear ODEs.

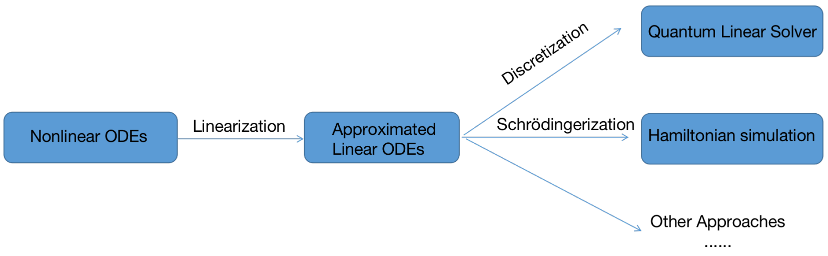


Figure 1. Quantum algorithms for nonlinear ODEs.

As highlighted in reference [22], the Carleman matrix is, in fact, a transposed finite-section matrix approximation of the Koopman Operator's generator on a polynomial basis. This specific connection and the weaker performance of Carleman Linearization in nonpolynomial cases motivate our research. Despite this connection, modern research on the Koopman Operator and Carleman Linearization follows distinct paths. Investigations into the Koopman Operator predominantly adopt a data-driven approach, as they use spatiotemporal data without requiring the explicit equation to find an approximation of the principal Koopman modes and, correspondingly, Koopman eigenpairs. In particular, numerical schemes such as dynamic mode decomposition (DMD) [23–25] and its diverse variations, including Extended Dynamic Mode Decomposition (EDMD) [26], make use of the data collected from time snapshots of a specified dynamical system. These approaches employ Singular Value Decomposition (SVD) and other matrix decomposition techniques to reveal the temporal and spatial–spectral attributes of the system. Expanding beyond DMD, EDMD, and its variants, methods for obtaining tractable representations have been further connected with deep neural networks [27,28]. Two specific neural network architectures have been explored in this context. One is based on the autoencoder principle, employing a low-dimensional latent space to enhance the interpretability of outcomes. The other architecture involves a higher-dimensional latent space, which often yields improved results when dealing with dynamical systems featuring continuous spectra. A comprehensive review of the theoretical framework and algorithms related to Data-Driven Koopman Operators is provided by [29].

Diverging from the previously discussed data-driven methods, our approach constitutes a progressive evolution of the linearization technique. This distinction mainly arises from our method of constructing the infinitesimal generator for the approximated Koopman operator. Rather than relying on data, we utilize the values from the right-hand side function of the ODEs at various discrete spatial points, thereby necessitating an explicit equation. More precisely, our method integrates the differentiation matrix in the spectral method [30,31], yielding an explicit matrix approximation of the Koopman Operator's generator. Subsequently, this matrix generates an explicit higher-dimensional linear equation that closely approximates the original nonlinear system. In this context, the identical approximation matrix was employed across various scalar observable configurations, with only the initial value being altered. This innovative approach was motivated by the spectral method for differential equations discussed in [32,33]. Therefore, like other spectral methods, our approach's effectiveness and convergence rate are influenced by the choice of the basis function. Potential options for the elements of D include polynomials [34], Fourier modes [32], radial basis functions [35], and spectral elements [36]. The best choice of basis functions is likely dependent on the specific characteristics of the underlying dynamical system and the data sampling strategy. Specifically, we adapt the Chebyshev Differentiation Matrix for demonstration purposes. This study focuses on autonomous systems and potentially serves as an alternative approach applicable in situations where Carleman Linearization has been conventionally utilized.

In addition to applications in quantum computing, our methods also hold potential for applications in traditional computational contexts. The most conventional method of linearization involves using a first-order Taylor expansion of the system dynamics. This technique, while proven effective and widely utilized in areas such as Model Predictive Control (MPC) and related domains, is limited to applications within a very narrow vicinity around the expansion point. To address these limitations more comprehensively, a range of more sophisticated linearization techniques such as Carleman Linearization [37] and the Koopman Operator [38] have been utilized. These methods were originally applied in the fields of dynamical systems and control [22].

The paper is structured as follows: Section 2 presents the background knowledge and discusses the Koopman Spectral Linearization Method in Section 3. In Section 4, we provide the numerical results, and the subsequent sections delve into the discussion and conclusions in Section 5.

2. Background

2.1. Carleman Linearization

Carleman Linearization provides a systematic methodology for deriving the corresponding lifted dynamics from an initial value problem defined as follows [37]:

$$\frac{d\boldsymbol{x}}{dt} = B_1\boldsymbol{x} + B_2\boldsymbol{x}^{\otimes 2} + \cdots + B_k\boldsymbol{x}^{\otimes k} \quad \text{with} \quad \boldsymbol{x}_0 = \boldsymbol{x}(t_0) \tag{1}$$

Here, the matrices $B_i \in \mathcal{R}^{d\times d^i}$ are time-independent, k is the degree of the polynomial ODE, and $\boldsymbol{x}^{\otimes i}$ denotes the i-fold tensor product for each positive integer i. We define an infinite-dimensional vector $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{y_3}, \ldots.)^T$ with $\boldsymbol{y_i} = \boldsymbol{x}^{\otimes i}$, and formulate matrices A^i_{i+j-1}

$$A^i_{i+j-1} = \sum_{v=1}^{i} \otimes_i B_j \tag{2}$$

where $\otimes_i B = I \otimes I \otimes I \otimes \cdots \otimes B \otimes \cdots \otimes I$ with B at the $(d-i+1)$-th position. Further, Formula (2) leads to the following equation:

$$\frac{d\boldsymbol{y_i}}{dt} = \sum_{j=1}^{k} A^i_{i+j-1}\boldsymbol{y_{i+j-1}} \tag{3}$$

Formally, this can be represented as an infinite-dimensional ordinary differential equation:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{\mathcal{A}}\boldsymbol{y} \qquad \text{with} \quad \boldsymbol{y_i}(t_0) = \boldsymbol{x}(t_0)^{\otimes i} \tag{4}$$

where $\boldsymbol{\mathcal{A}}$ is an infinite-dimensional matrix defined as:

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} A^1_1 & A^1_2 & A^1_3 & \ldots & A^1_k & 0 & 0 & \ldots \\ 0 & A^2_2 & A^2_3 & \ldots & A^2_k & A^2_{k+1} & 0 & \ldots \\ 0 & 0 & A^3_3 & \ldots & A^3_k & A^3_{k+1} & A^3_{k+2} & \ldots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \end{bmatrix} \tag{5}$$

In practice, the system is often truncated at a certain order N. Here is an illustrative example [37] of Carleman Linearization applied to the following system:

$$\frac{dx}{dt} = x^2 \tag{6}$$

The Carleman Linearization procedure can derive a linear ODE-like

$$\frac{d}{dt}\begin{bmatrix} x \\ x^2 \\ x^3 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 2 & 0 & 0 & \dots \\ 0 & 0 & 0 & 3 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}\begin{bmatrix} x \\ x^2 \\ x^3 \\ \vdots \end{bmatrix} \tag{7}$$

2.2. Koopman Operator

Firstly, consider a d-dimension (i.e., $\boldsymbol{x} \in X \subset \mathbb{R}^d$) dynamical system described by first-order autonomous order ordinary differential equations with $t \in [t_0, T]$:

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{8}$$

where $\boldsymbol{x} = (x_1, x_2, \dots, x_d)^T$ belongs to a space X, and this dynamics $\boldsymbol{f}$ is also a d-dimensional vector valued function with $f = (f_1, f_2, \dots, f_d)^T$. For our analysis, we introduce observables or measurement functions, represented by the function $g : X \to \mathbb{R}$; note that this is a function on $\mathcal{G}(X)$ (e.g., an $\mathcal{L}^2$ Hilbert space). And the flow map F_t represents the evolution of the system dynamics as a mapping on X:

$$F_t : \boldsymbol{x}(t_0) \to \boldsymbol{x}(t + t_0) \tag{9}$$

The Koopman operator family, denoted as $\{\mathcal{K}_t\}$, is defined as:

$$\mathcal{K}_t : g(\boldsymbol{x}(t_0)) \to g(F_t(\boldsymbol{x}(t_0))) = g(\boldsymbol{x}(t_0 + t)) \tag{10}$$

Alternatively, we can express $\mathcal{K}_t$ as a composition of functions:

$$\mathcal{K}_t : g \to g \circ F_t \tag{11}$$

It is crucial to note that the Koopman operator is linear, owing to the linearity of the function space $\mathcal{G}(X)$. Further, we can define the generator of the Koopman operator $\mathcal{K}$

$$\mathcal{K}g := \lim_{t \to 0} \frac{\mathcal{K}_t g - g}{t} = \frac{\partial g}{\partial t} \tag{12}$$

The generator can be explicitly formulated as follows:

$$\mathcal{K}g = \frac{\partial g}{\partial t} = \nabla g(\boldsymbol{x})\frac{d\boldsymbol{x}}{dt} = \nabla g(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}) \tag{13}$$

The transformation of representing it in the form of an operator can be further extended as follows:

$$\mathcal{K} = \nabla \cdot \boldsymbol{f}(\boldsymbol{x}) = \sum_{i=1}^{d} f_i(x)\frac{\partial}{\partial x_i} \tag{14}$$

As a trade-off, we convert a nonlinear mapping on the variable x into a linear operator on the variable g, which leads to an infinite-dimensional space.

In Koopman Spectral Theory, eigenvalues and eigenfunctions (or eigenpairs for brevity) are employed for practical numerical computations [24,39]. Suppose $(\lambda, \phi(\boldsymbol{x}))$ is an eigenpair for the generator $\mathcal{K}$. By definition,

$$\mathcal{K}\phi(\boldsymbol{x}(t + t_0)) = \frac{d\phi(\boldsymbol{x}(t + t_0))}{dt} = \lambda\phi(\boldsymbol{x}(t + t_0)) \tag{15}$$

Based on the ODE in Equation (15), we can derive the following outcomes:

$$\phi(\boldsymbol{x}(t+t_0)) = \phi(\boldsymbol{x}(t_0))e^{\lambda t} \tag{16}$$

Actually, $\phi(\boldsymbol{x})$ is also an eigenfunction for $\mathcal{K}_t$; this can be observed from

$$\mathcal{K}_t\phi(\boldsymbol{x}(t+t_0)) = \phi(\boldsymbol{x}(t_0+2t)) = e^{\lambda t}\phi(\boldsymbol{x}(t_0+t)) \tag{17}$$

Therefore, if we denote $e^{\lambda t}$ as μ, then $(\mu, \phi) = (e^{\lambda t}, \phi)$ is an eigenpair for $\mathcal{K}_t$.

If $g \in \mathcal{G}(X) \subset Span\{\phi_k\}$, then even the nonlinear observable can be represented as a linear combination of eigenfunctions. This can be formulated as $g = \sum_j c_j\phi_j$, where $c_j \in \mathbb{R}$ is the corresponding coefficient of g concerning ϕ_j. Further, as pointed out in [39], the evolution of the observable can also be represented as a linear combination of eigenpairs which is derived from:

$$\mathcal{K}_t g(\boldsymbol{x}(t+t_0)) = \mathcal{K}_t(\sum_{j=1}^{\infty} c_j\phi_j(\boldsymbol{x}(t_0+t))) = \sum_{j=1}^{\infty} c_j e^{\lambda_j t}\phi_j(\boldsymbol{x}(t_0+t)) = \sum_{j=1}^{\infty} c_j\mu_j\phi_j(\boldsymbol{x}(t_0+t)) \tag{18}$$

Hence, we can infer

$$g(\boldsymbol{x}(t+t_0)) = \sum_{j=1}^{\infty} c_j e^{\lambda_j t}\phi_j(\boldsymbol{x}(t_0)) \tag{19}$$

Multi-dimensional observables are based in the same manner. Just consider vector-valued c_j with the same dimension of the observable. c_j is also known as the j-th Koopman mode [29].

3. Koopman Spectral Linearization Method

3.1. Construction of the Lifted Matrix

By adopting a construction of the matrix approximation from [30,31], the generator of the Koopman Operator can be approximated as follows.

Currently, when d = 1, for some eigenpairs (ϕ, λ) based on Equation (7),

$$\mathcal{K}\phi = f \cdot \frac{\partial}{\partial x} \cdot \phi \tag{20}$$

To derive a finite-dimensional approximation, we start from a polynomial interpolation of the eigenfunction $\phi(x)$ on spatial discretized Gauss–Lobatto points $\{\xi_i\}_{i=1}^N$, which gives us

$$\phi(x) \approx \phi^N(x) = \sum_{i=1}^{N} \phi^N(\xi_i)L_i(x) \tag{21}$$

where L_j are Lagrange polynomials such that $L_j(\xi_i) = \delta_{ij}$, where δ_{ij} is the delta function. Therefore, we obtain

$$K \begin{bmatrix} \phi^N(\xi_0) \\ \phi^N(\xi_1) \\ \vdots \\ \phi^N(\xi_N) \end{bmatrix} = \begin{bmatrix} f(\xi_0) & & & \\ & f(\xi_1) & & \\ & & \ddots & \\ & & & f(\xi_N) \end{bmatrix} D \begin{bmatrix} \phi^N(\xi_0) \\ \phi^N(\xi_1) \\ \vdots \\ \phi^N(\xi_N) \end{bmatrix} \tag{22}$$

Based on Equation (22), the finite representation of K is

$$K = diag\{f(\xi_0), f(\xi_1), \dots, f(\xi_N)\}D, \tag{23}$$

where *diag* creates a diagonal matrix with the main diagonal elements. When $d = 2$, let $\{\xi_i\}_{i=1}^N$ and $\{\eta_j\}_{j=1}^N$ be the Gauss–Lobatto points near x_1 and x_2. And denote $\Theta = \{(\xi_i, \eta_j)\}_{i,j=1}^N$. Now, for every bivariate eigenfunction $\phi(x_1, x_2)$, we have polynomial interpolation ϕ^N

$$\phi(x_1, x_2) \approx \phi^N(x_1, x_2) = \sum_{i=1}^{N}\sum_{j=1}^{N} \phi^N(\xi_i, \eta_j) L_i(x_1) L_j(x_2) \tag{24}$$

Hence, all function values on the collocation points can be formed as a matrix denoted as $\phi^N(\mathbf{\Theta})$

$$\phi^N(\mathbf{\Theta}) = \begin{bmatrix} \phi^N(\xi_1,\eta_1) & \phi^N(\xi_1,\eta_2) & \dots & \phi^N(\xi_1,\eta_N) \\ \phi^N(\xi_2,\eta_1) & \phi^N(\xi_2,\eta_2) & \dots & \phi^N(\xi_2,\eta_N) \\ \vdots & \vdots & \ddots & \vdots \\ \phi^N(\xi_N,\eta_1) & \phi^N(\xi_N,\eta_2) & \dots & \phi^N(\xi_N,\eta_N) \end{bmatrix} \tag{25}$$

Let D_1, D_2 be the differentiation matrices for x_1 and x_2, respectively, and $f_1(\mathbf{\Theta})$,$f_2(\mathbf{\Theta})$ be the matrices of f_1, f_2 evaluated at (ξ_i, η_j). And we denote $\mathcal{K}\phi^N(\mathbf{\Theta})_{ij} = \mathcal{K}\phi^N(\xi_i, \eta_j)$. Then, the formula below can be derived based on (7):

$$\mathcal{K}\phi^N(\mathbf{\Theta}) = f_1(\mathbf{\Theta}) \odot (D_1\phi^N(\mathbf{\Theta})) + f_2(\mathbf{\Theta}) \odot (\phi^N(\mathbf{\Theta})D_2^T) \tag{26}$$

In the equation above, $\odot$ means the Hadamard product. We vectorize all the matrix operations as vector operations via *vec*, which flatten a matrix into a column vector; this process can be formulated as:

$$\begin{aligned} \mathcal{K}vec(\phi^N(\mathbf{\Theta})) &= vec(f_1(\mathbf{\Theta})) \odot ((I \otimes D_1)vec(\phi^N(\mathbf{\Theta}))) + vec(f_2(\mathbf{\Theta})) \odot ((D_2 \otimes I)vec(\phi^N(\mathbf{\Theta}))) \\ &= [diag(vec(f_1(\mathbf{\Theta})))(I \otimes D_1) + diag(vec(f_2(\mathbf{\Theta}))(D_2 \otimes I))]vec(\phi^N(\mathbf{\Theta}) \\ &= Kvec(\phi^N(\mathbf{\Theta}) \end{aligned} \tag{27}$$

Therefore, for $d = 2$ cases, the construction of matrix K is given by

$$K = diag(vec(f_1(\mathbf{\Theta})))(I \otimes D_1) + diag(vec(f_2(\mathbf{\Theta}))(D_2 \otimes I)) \tag{28}$$

In general, for a d-dimensional case, consider Gauss–Lobatto points for each coordinate as $\{\xi^1_{i_1}\}_{i_1=1}^N, \{\xi^2_{i_2}\}_{i_2=1}^N, \dots, \{\xi^d_{i_d}\}_{i_d=1}^N$. And use Θ to denote the collection of all the collocation points, i.e., $\Theta = \{(\xi^1_{i_1}, \xi^2_{i_2}, \dots, \xi^d_{i_d}) : i_1, i_2, \dots, i_d, \text{traverse from 1 to N}\}$. And the eigenfunction ϕ is approximated by

$$\phi(x_1, \dots, x_d) \approx \phi^N(x_1, \dots, x_d) = \sum_{i_1,\dots,i_d=1}^{N} \phi^N(\xi^1_{i_1}, \xi^2_{i_2}, \dots, \xi^d_{i_d}) L_{i_1}(x_1) \dots L_{i_d}(x_d) \tag{29}$$

Correspondingly, $\phi^N(\mathbf{\Theta})$ is a tensor of eigenfunction values on collocation points, and $D_1, D_2, \dots D_d$ means differentiation matrices. With n-mode multiplication in tensor algebra, we can write

$$\mathcal{K}\phi^N(\mathbf{\Theta}) = \sum_{i=1}^{d} f_i(\Theta) \odot (\phi^N(\mathbf{\Theta}) \times_i D_i) \tag{30}$$

Pursuing the analogous vectorization approach in Equation (27), we can reformulate Equation (30) as

$$\begin{aligned} \mathcal{K}vec(\phi^N(\mathbf{\Theta})) &= \{\sum_{i=1}^{d} vec(f_i(\mathbf{\Theta})) \odot (\otimes_i D_i)\}vec(\phi^N(\mathbf{\Theta})) \\ &= \{\sum_{i=1}^{d} diag(vec(f_i(\mathbf{\Theta})))(\otimes_i D_i)\}vec(\phi^N(\mathbf{\Theta})) \end{aligned} \tag{31}$$

Consequently,

$$K = \sum_{i=1}^{d} diag(vec(f_i(\boldsymbol{\Theta})))(\otimes_i D_i) \tag{32}$$

where $\otimes_i D_i = I \otimes I \otimes I \otimes \cdots \otimes D_i \otimes \cdots \otimes I$ with D_i at the $(d - i + 1)$-position. To be more intuitive, here is an example when $d = 3$:

$$\begin{aligned} K =& diag\{vec(F_1)\} I \otimes I \otimes D_1 \\ &+ diag\{vec(F_2)\} I \otimes D_2 \otimes I \\ &+ diag\{vec(F_3)\} D_3 \otimes I \otimes I \end{aligned} \tag{33}$$

3.2. Solution of the Linear System

Now, let us examine the solution of the linear system to provide further insight into why it serves as an approximation of the original nonlinear one. Referring back to Equation (19), we have

$$g(\boldsymbol{x}(t)) \approx g^N(\boldsymbol{x}(t)) = \sum_{j=1}^{N} \hat{c}_j \phi_j^N(\boldsymbol{x}(t_0)) e^{\hat{\lambda}_j} \tag{34}$$

When $d = 1$, for a scalar observable g, consider Gauss–Lobatto points on a region with radius r around x_0, where $\Theta = \{\xi_i\}_{i=1}^N$ and $\xi_1 < \xi_2 < \cdots < \xi_N$. Here, we pick the odd number N such that $\xi_{(N+1)/2} = x_0 = x(t_0)$. Thus, all Gauss–Lobatto points are on the interval $[x_0 - r, x_0 + r]$. Suppose $(\hat{\lambda}_j, v_j)$ is an eigenpair of matrix K. Vector v_j are used to approximate eigenfunction ϕ_j^N evaluated at the collocation points. We have $(v_j)_i = \phi_j^N(\xi_i) \approx \phi_j(\xi_i)$. Hence,

$$g^N(x(t)) = \sum_{j=1}^{N} \hat{c}_j \phi_j^N(x(t_0)) e^{\hat{\lambda}_j t} = \sum_{j=1}^{N} \hat{c}_j \phi_j^N(\xi_{(N+1)/2}) e^{\hat{\lambda}_j t} = \sum_{j=1}^{N} \hat{c}_j (v_j)_{\frac{N+1}{2}} e^{\hat{\lambda}_j t} \tag{35}$$

Now consider the N-dimensional linear system below

$$\frac{dy}{dt} = Ky \qquad y_0 = (g(x_0 - r), \ldots, g(x_0 + r))^T \tag{36}$$

We can write the analytical solution

$$y(t) = e^{Kt} y_0 \tag{37}$$

Suppose that the eigendecomposition of $K = V \Lambda V^{-1}$, where each column of V is an eigenvector of K denoted as v_j, and Λ contains all the eigenvalues. Further, by setting $t = 0$ in Equation (35), we have

$$g_N(x_0) = \sum_{j=1}^{N} \hat{c}_j \phi_j^N(x_0) \tag{38}$$

This is also true for different initial values. Thus,

$$g_N(\xi_i) = \sum_{j=1}^{N} \hat{c}_j \phi_j^N(\xi_i) = \sum_{j=1}^{N} \hat{c}_j (v_j)_i, \quad i = 1, 2, \ldots, N \tag{39}$$

Thus, we have $V^{-1}y_0 = c$ since $y(0) = (g^N(\xi_1), g^N(\xi_2), \dots, g^N(\xi_N))^T$, where c is called the Koopman mode:

$$\begin{aligned} y(t) &= e^{Kt}y_0 \\ &= e^{V\Lambda t V^{-1}}y_0 \\ &= Ve^{\Lambda t}V^{-1}y_0 \\ &= [v_1, v_2, \dots, v_N] \begin{bmatrix} e^{\lambda_1 t} & & & \\ & e^{\lambda_2 t} & & \\ & & \ddots & \\ & & & e^{\lambda_N t} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \\ &= \sum_{j=1}^{N} c_j e^{\lambda_j t} v_j \end{aligned} \tag{40}$$

And $g^N(x(t)) = \sum_{j=1}^{N} c_j e^{\lambda_j t}(v_j)_{\frac{N}{2}}$.

Notice that c_j are Koopman modes, and $e^{\lambda_j t}$ are eigenfunctions. In the general case, for the d-dimensional nonlinear system, suppose $\mathbf{\Theta}$ represents d-dimensional collocation points, and K is constructed as Equation (32). Now consider the linear system given below

$$\frac{d\boldsymbol{y}(t)}{dt} = K\boldsymbol{y} \qquad \boldsymbol{y}(0) = vec(g(\mathbf{\Theta})). \tag{41}$$

Again, by vectorization, $\phi_j(\boldsymbol{x_0})$ can be approximated by the "middle term" of tensor $\phi^N(\mathbf{\Theta})$, which leads to

$$y(t) = \sum_{j=1}^{N^d} c_j e^{\lambda_j t}(v_j). \tag{42}$$

Only the middle term is needed for this long vector to compute the observable:

$$g_N(x(t)) = \sum_{j=1}^{N^d} c_j e^{\lambda_j t}(v_j)_{\frac{N^d+1}{2}}, \tag{43}$$

which is exactly the solution of (41).

Unlike Carleman Linearization, which captures the information of all variables within a single linear ordinary differential equation, our linearization approach is primarily tailored for scalar observables. When we need to compute multi-dimensional or distinct scalar observables, we can still employ the previously constructed lifted matrix K. However, this requires transforming the initial conditions accordingly. For instance, when computing an observable $g(\boldsymbol{x}) = \boldsymbol{x} = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), g_3(\boldsymbol{x}))$, we would need to utilize three different initial conditions: $g_1(\mathbf{\Theta}), g_2(\mathbf{\Theta}), g_3(\mathbf{\Theta})$ This adaptation allows us to extend the applicability of our approach to scenarios involving diverse observable functions since K can be reused.

As for the solution of Carleman Linearization, the matrix $\boldsymbol{\mathcal{A}}$ shall be truncated at a certain level N, which yields the $\boldsymbol{\mathcal{A}_N}$ matrix as follows:

$$\boldsymbol{\mathcal{A}}_N = \begin{bmatrix} A_1^1 & A_2^1 & A_3^1 & \dots & A_k^1 & 0 & 0 & \dots \\ 0 & A_2^2 & A_3^2 & \dots & A_k^2 & A_{k+1}^2 & 0 & \dots \\ 0 & 0 & A_3^3 & \dots & A_k^3 & A_{k+1}^3 & A_{k+2}^3 & \dots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \dots. & 0 & 0 & 0 & A_N^N \end{bmatrix}. \tag{44}$$

Then, the infinite linear system can be approximated by

$$\frac{dy}{dt} = \boldsymbol{\mathcal{A}}_N y \tag{45}$$

and the approximated solution of an original nonlinear system is exactly the first d elements of the solution for Equation (45).

4. Numerical Result

This section presents the performance comparison of Koopman Spectral Linearization against Carleman Linearization on five nonlinear dynamic models in Section 4.1. In each example, we visualized the curves of specific solution components and investigated the influence of truncation order N on accuracy. As N increases, both the precision of linearization and the computational effort increase correspondingly. In practical applications, the selection of parameter N should be based on the required accuracy and available computational resources. Since Carleman Linearization can only be applied to polynomial cases, all the following nonpolynomial examples are converted to polynomial form using higher-order Taylor expansions before computation. The reference solution is generated by The 4th order Runge–Kutta method (RK4) is used if no closed-form solution is available. We also aim to compare the loss of accuracy in the linearization step under the assumption of an ideal quantum computing environment, where a high-precision quantum linear solver or other quantum algorithms for linear ODEs are available. Hence, the solutions of the resulting linear ODEs are directly computed using the `exp` function in MATLAB 2022a. Next, in Section 4.2, we further compare the matrix size and computational cost (all the MATLAB codes can be downloaded at https://github.com/DongDongShiShi/Koopman-Comparison (accessed on 30 May 2024).

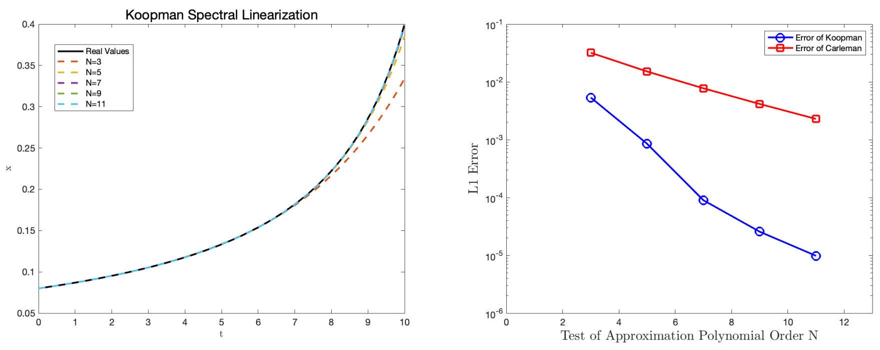
4.1. Linearize Dynamics with Koopman Spectral Linearization

4.1.1. Quadratic Model

We start with a simple nonlinear ODE, and the governing ODE is given by

$$\frac{dx}{dt} = x^2$$

We set $x(0) = 0.08$ and T = 10 in this example. This quadratic model has a closed form solution $x(t) = \frac{1}{(1/x_0)-t}$. In the tests, we fix $r = 0.03$. Figure 2 summarizes the results of the first comparison. Figure 2b shows the exponential convergence of our approach concerning the truncation order, which is similar to the conclusion in the conventional Carleman Linearization method. Further, our approach shows a faster convergence rate.



(**a**) Solutions with different N (**b**) Test of Truncation Order

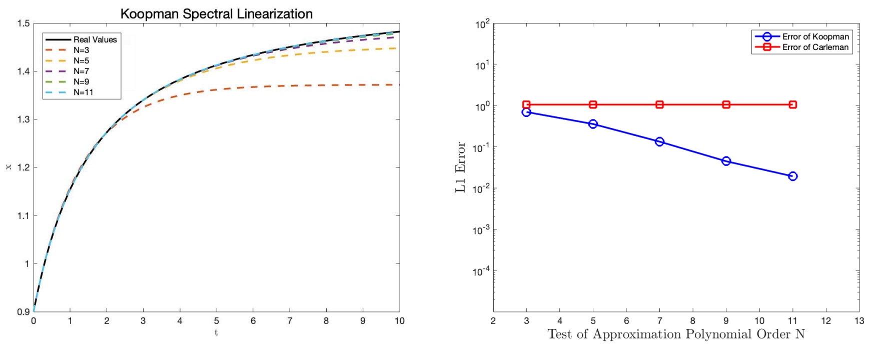
Figure 2. Quadratic model: (**a**) solutions by the Koopman Linearization with different N; (**b**) the L_1 error of the linearized ODE with truncation order N.

4.1.2. Cosine Square Model

This cosine square model is a synthetic model invented as a nonpolynomial case for our demonstrative purposes. The governing equation is given as

$$\frac{dx}{dt} = \cos^2(x).$$

And we set $x(0) = 0.1$, $T = 10$. Despite the nonlinear nature, we still have a closed-form solution $x(t) = \arctan(-0.5t + \tan(x_0))$. We fix $r = 0.3$ in the tests. It is important to note that as the system evolves over time, errors gradually accumulate. This is particularly evident when using lower-order polynomial interpolation (i.e., $N = 3$). Specifically, in Figure 3a, beyond $t = 8$, the curve begins to exhibit a more pronounced deviation from the standard solution. Figure 3b shows the exponential convergence of our approach concerning the truncation order, which is still similar to the conclusion in the previous example. Conversely, for this nonpolynomial model, the accuracy of Carleman Linearization is very low, and we did not observe any significant changes in accuracy across the different values of N.



(**a**) Test of Solution (**b**) Test of Truncation Order

Figure 3. Cosine square model: (**a**) solutions by Koopman Linearization with different N; (**b**) the L_1 error of the linearized ODE with truncation order N.

4.1.3. Simple Pendulum Model

The simple pendulum is a well-studied model in science and engineering. The movement of the pendulum is described by the following second-order ordinary different equation:

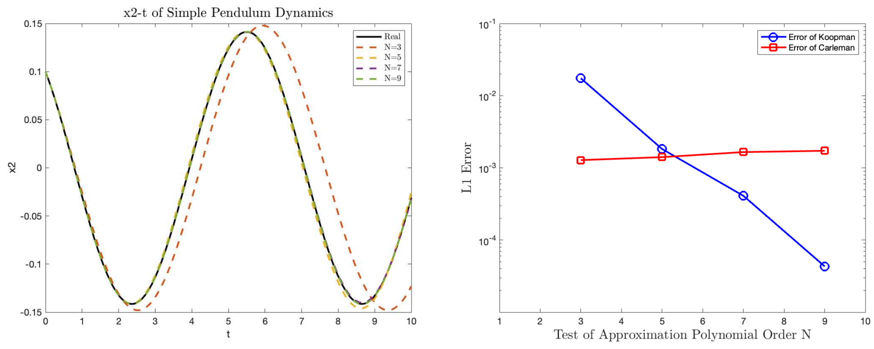
$$\frac{d^2\theta}{dt} = -\frac{\text{g}}{\text{L}}\sin(\theta),$$

where L is the length of the pendulum, θ is the displacement angle, and the parameter g is the gravity acceleration. This second-order equation can be converted to a two-dimensional first-order ODE system. As a notation, we define $x_1 = \theta$ and $x_2 = \frac{d\theta}{dt}$. Also, we set $L = g = 9.8$ for simplicity. Correspondingly,

$$\frac{dx_1}{dt} = x_2,$$
$$\frac{dx_2}{dt} = -\sin(x_1),$$

and we set $x(0) = (0.1, 0.1)^T$, $T = 5$. We fix $r = 1$ in the tests.

Figure 4 presents the numerical result of a simple pendulum. Similar to the previous result, the exponential convergence of error concerning truncation order N is observed from Figure 4b. Although the accuracy of Carleman Linearization for this nonpolynomial model is significantly higher compared to its application on the cosine square model and even more accurate than Koopman Spectral Linearization at $N = 3$ and $N = 5$, the accuracy does not significantly change as N increases.



(**a**) Test of Solution (**b**) Test of Truncation Order

Figure 4. Simple pendulum model: (**a**) solutions by Koopman Linearization with different N; (**b**) the L_1 error of the linearized ODE with truncation order N.

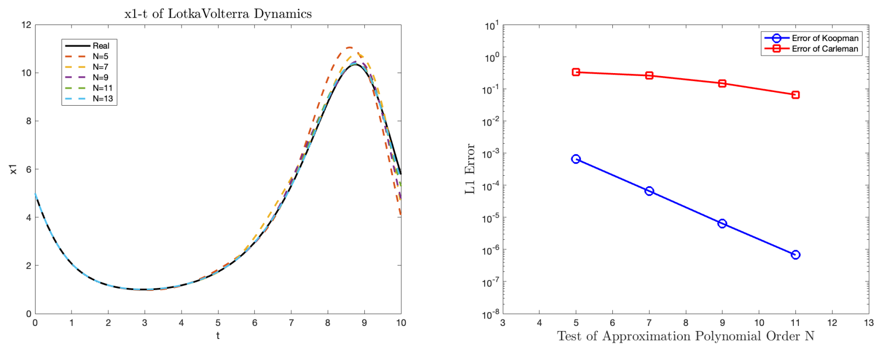
4.1.4. Lotka–Volterra Model

The Lotka–Volterra model, also known as the predator–prey model [40], is a mathematical model used to describe the interactions between predators and prey in an ecosystem. The model describes the interaction between two fundamental populations: prey and predator populations. Here, we define as below:

$$\frac{dx_1}{dt} = 1.1x_1 - 0.4x_1x_2,$$
$$\frac{dx_2}{dt} = 0.1x_1x_2 - 0.4x_2.$$

We set $x(0) = (5,5)^T$, $T = 10$ and $r = 3$.

Figure 5 presents the results of these tests for the Lotka–Volterra model. Again, errors are decreased exponentially with respect to the truncation order. For this polynomial model, the accuracy of Carleman Linearization improves with increasing N. However, regarding both numerical accuracy and convergence speed, Koopman Spectral Linearization shows a significant advantage.



(**a**) Test of Solution (**b**) Test of Truncation Order

Figure 5. Lotka–Volterra model: (**a**) solutions by Koopman Linearization with different N; (**b**) the L_1 error of the linearized ODE with truncation order N.

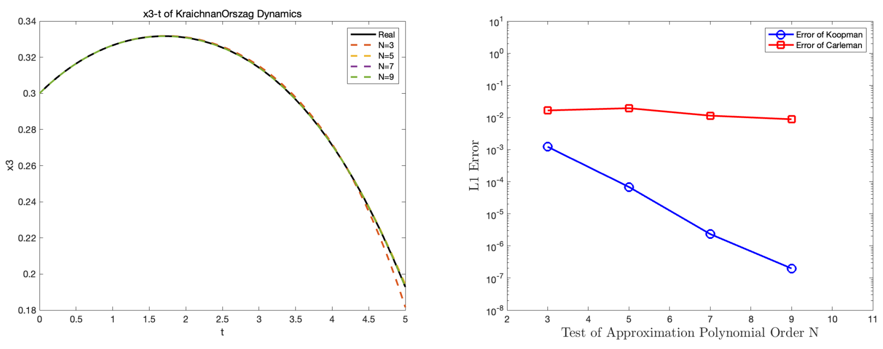
4.1.5. Kraichnan–Orszag Model

The Kraichnan–Orszag model was presented in [41] for modeling fluid dynamics. This three-dimensional nonlinear model is given by

$$\begin{aligned}\frac{dx_1}{dt} &= x_2x_3,\\ \frac{dx_2}{dt} &= x_1x_3,\\ \frac{dx_3}{dt} &= -2x_1x_2.\end{aligned}$$

We set $x(0) = (0.1, -0.2, 0.3)^T$, $T = 5$ and $r = 0.1$.

Figure 6 presents the results of the Kraichnan–Orszag model. Different from the previous example, a shorter time period is set due to the strong oscillations of the Kraichnan–Orszag model.



(**a**) Test of Solution (**b**) Test of Truncation Order

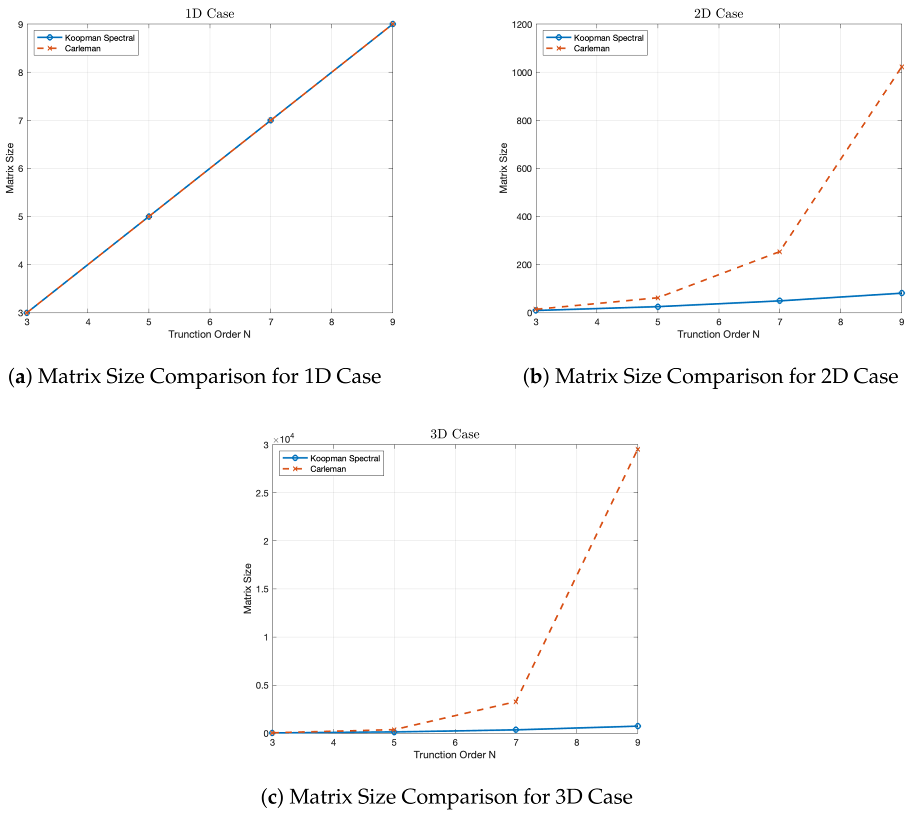
Figure 6. Kraichnan–Orszag model: (**a**) solutions by Koopman Linearization with different N; (**b**) the L_1 error of the linearized ODE with truncation order N.

The truncation order testing figures illustrate the error as a function of the truncation order N and provide a comparison between Carleman Linearization and our method. The mistake of Carleman Linearization in polynomial cases (i.e., Sections 4.1.1, 4.1.4 and 4.1.5) exhibits exponential decay, a phenomenon that has been substantiated by recent theoretical research [42,43]. Similarly, our method leverages the Chebyshev node interpolation approach, which also enjoys exponential convergence guarantees and demonstrates similar exponential convergence in numerical experiments.

For all the polynomial examples in Sections 4.1.1, 4.1.4 and 4.1.5, it is evident that our approach exhibits higher accuracy at the same truncation order, and the error decreases more rapidly as the truncation order is increasing. Moreover, when applied to nonpolynomial dynamical systems, our method showcases superior numerical accuracy and faster convergence rates. In Sections 4.1.2 and 4.1.3, we present a comparative analysis of the error–truncation order relationship in nonpolynomial dynamical systems. Both nonpolynomial dynamics are subjected to 12th-order Taylor expansions to transform them into polynomial forms. The Carleman method exhibits significantly lower accuracy in this scenario, with notably slower convergence. This limitation arises because in nonpolynomial cases, the relevant part of the Taylor expansion is confined to orders lower than the truncation order, resulting in substantial errors when employing relatively small expansion orders as demonstrated in our study. Consequently, our method demonstrates heightened precision in nonpolynomial cases. Even in the case of the 'simple pendulum' where $N = 3$, our method initially exhibits lower accuracy. However, as N increases to 9, the errors obtained by our method rapidly become dramatically smaller than those produced by the Carleman method.

4.2. Matrix Size and Computational Cost Comparison

Following the construction procedure introduced in Section 3, we apply the tensor product rule to construct collocation points. Consequently, the approximation matrix of the Koopman generator has a size of $N^d \times N^d$. However, we can reuse the lifted matrix with different initial values for multi-dimensional dynamics. In the case of the Carleman Linearization procedure, the matrix size will be $\sum_{i=1}^{N} d^i \times \sum_{i=1}^{N} d^i$, and we can approximate the size of the Carleman lifted matrix as $O(d^N)$. Therefore, when considering the matrix size of the derived linear system with a relatively small truncation order N, the matrix in our approach can be significantly smaller than the matrix in Carleman Linearization. To further illustrate this point, Figure 7 compares our approach with the size of the Carleman Linearization matrix.



(**a**) Matrix Size Comparison for 1D Case (**b**) Matrix Size Comparison for 2D Case

(**c**) Matrix Size Comparison for 3D Case

Figure 7. Matrix size comparison of Koopman Spectral and Carleman Linearizations.

While our approach generates d linear systems with the same matrix but different initial values, which may not be as convenient as Carleman Linearization, where only one linear system is produced, the significant difference in matrix sizes results in a notable increase in computational cost. As an illustrative example, we present the running time and error in Table 1 for a truncation order of $N = 9$. The computational time required for Carleman Linearization is slower than our approach, especially when $d = 2$ or 3. In particular, for the Kraichnan–Orszag model ($d = 3$), the Carleman Linearization procedure took 32.0546 s, being considerably slower than our approach.

On the other hand, our method's capacity to yield numerical results with smaller matrices, reduced computational time, and increased accuracy, all under the same truncation order, hinges upon a critical prerequisite—namely, the selection of an appropriate parameter r, a requirement not present in the Carleman Linearization method. Furthermore, when only the time-domain evolution differential equations of the dynamical system are available in advance, without access to spatial evolution information of the state, our method cannot capture the state's spatial evolution. Therefore, unless supplementary information is accessible regarding the spatial evolution of the dynamical system, allowing for a reasonable estimation of r and surpassing the precision of Carleman Linearization remains unattainable.

Table 1. Error and time cost compared with Carleman Linearization ($N = 9$).

	Error		Time (s)	
	Carleman	**Koopman**	**Carleman**	**Koopman**
Quadratic	4.19×10^{-3}	2.56×10^{-5}	0.0147	0.0089
Lotka-Volterra	1.49×10^{-1}	6.39×10^{-6}	0.1525	0.0188
Kraichnan-Orszag	8.80×10^{-3}	1.99×10^{-7}	32.0546	0.1370
Cosine Square	1.26	7.39×10^{-4}	0.0059	0.0049
Simple Pendulum	1.20×10^{-3}	7.02×10^{-5}	0.0368	0.0141

5. Discussion and Conclusions

The Koopman Spectral Linearization method explicitly uses the differentiation matrix to derive the lifted matrix of nonlinear autonomous dynamical systems. It provides a finite-dimensional representation for the generator of the Koopman Operator with a polynomial basis. Therefore, similar to Carleman Linearization, our approach offers a linearization method for nonlinear autonomous ordinary differential equations. In each numerical experiment presented, our approach exhibits exponential convergence with respect to truncation order N, just like Carleman Linearization. Under the same truncation order, Koopman Spectral Linearization tends to exhibit significantly higher accuracy associated with lower time cost compared to Carleman Linearization, especially when the dynamics are not polynomial. Therefore, it is more suitable as an alternative linearization method where high-accuracy approximations are needed and f is nonpolynomial. Different from Carleman Linearization, which describes the evolution of the state itself, our approach is used to describe a scalar smooth observable, which is more flexible.

However, despite its advantages, the Koopman Spectral Linearization method has some limitations. It is essential to consider certain drawbacks when evaluating its applicability since a reasonable estimation of radius r is necessary to obtain an accurate approximation of the eigenfunctions. We employ interpolation within a local space of radius r around the initial point x_0 to approximate the eigenfunctions across in a local domain. However, since the spatial range of the system evolution during the process is not known in advance, it is very challenging to select the value of r without additional information; when theoretical guidance is unavailable, r is chosen empirically as a relatively small value (i.e., 0.1–0.5) to ensure a higher precision of approximation. If the spatial range of the system's evolution is known beforehand, such as the case of fluid moving in a pipeline, r can then be selected based on spatial constraints. To further understand how this parameter r influences the accuracy, a comprehensive numerical analysis will be included in future work. In addition, the global interpolation method might overcome the limitation of the estimate r. Besides concerns regarding the parameter r, the degree of nonlinearity in physical systems can vary significantly, which might also influence the performance of our linearization method. Particularly for highly nonlinear systems such as turbulence, no linearization technique is capable of providing high-precision linear approximations over extended periods and large areas. In such instances, data-driven approaches may offer more satisfactory solutions, addressing the complexities inherent in these systems more effectively.

Regarding the matrix size as indicated in Section 4.2, the Koopman Spectral Method obtains a much smaller matrix with even higher accuracy compared with Carleman Linearization. However, the matrix size is exponential increased with respect to dimension d (i.e., N^d) since the tensor product rule is applied. Therefore, both Carleman Linearization (i.e., $O(d^N)$) and our approach might not be efficient for the relatively high-dimensional problems on classic computers. A possible way to overcome this limitation is to adapt sparse grid methods to construct collocation points, which has been found successful in [30]. As pointed out in the Koopman Operator Theory, Carleman Linearization and our approach rely on tensor product rule; therefore, they need further modification to deal with high-dimensional problems on classical computers. However, on quantum computers, tensor products may not be a challenge, which may lead to innovative designs of algorithms that are completely different from their counterpart on classical computers. This is the main reason we conduct these comparisons.

As previous sections noted, the underlying principle (Schrödinger equation) of quantum computers is linear. Consequently, there are fundamental difficulties in using quantum computing to solve nonlinear ODEs/PDEs. Currently, the linearization of differential equations appears to be an unavoidable step in addressing this challenge. In particular, Carleman Linearization as a solution strategy has garnered significant attention in recent research [17]. By reviewing the underlying theory of Carleman Linearization—Koopman Operator Theory—and integrating it with the concept of differentiation matrices in spectral methods, we propose the Koopman Spectral Linearization method. Our novel linearization approach might be useful when designing quantum algorithms for nonlinear ODEs/PDEs under certain situations.

Author Contributions: Conceptualization, X.Y.; methodology, X.Y. and D.S.; software, D.S.; validation, D.S.; writing—original draft preparation, X.Y. and D.S.; project administration, X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Foundation (NSF) under Grant No. 2143915.

Data Availability Statement: Inquiries about data availability should go directly to the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
QLSA	Quantum Linear System Algorithm
DMD	Dynamic Mode Decomposition
EDMD	Extended Dynamic Mode Decomposition
SVD	Singular Value Decomposition

References

1. Łukaszewicz, G.; Kalita, P. *Navier–Stokes Equations*; Advances in Mechanics and Mathematics; Springer: Cham, Switzerland, 2016.
2. Cooper, I.; Mondal, A.; Antonopoulos, C.G. A SIR model assumption for the spread of COVID-19 in different communities. *Chaos Solitons Fractals* **2020**, *139*, 110057. [CrossRef]
3. Merton, R.C. Applications of option-pricing theory: Twenty-five years later. *Am. Econ. Rev.* **1998**, *88*, 323–349.
4. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef]
5. Berry, D.W. High-order quantum algorithm for solving linear differential equations. *J. Phys. Math. Theor.* **2014**, *47*, 105301. [CrossRef]
6. Berry, D.W.; Childs, A.M.; Ostrander, A.; Wang, G. Quantum algorithm for linear differential equations with exponentially improved dependence on precision. *Commun. Math. Phys.* **2017**, *356*, 1057–1081. [CrossRef]
7. Montanaro, A.; Pallister, S. Quantum algorithms and the finite element method. *Phys. Rev. A* **2016**, *93*, 032324. [CrossRef]
8. Childs, A.M.; Liu, J.P. Quantum spectral methods for differential equations. *Commun. Math. Phys.* **2020**, *375*, 1427–1457. [CrossRef]
9. Linden, N.; Montanaro, A.; Shao, C. Quantum vs. classical algorithms for solving the heat equation. *Commun. Math. Phys.* **2022**, *395*, 601–641. [CrossRef]
10. Engel, A.; Smith, G.; Parker, S.E. Quantum algorithm for the Vlasov equation. *Phys. Rev. A* **2019**, *100*, 062315. [CrossRef]
11. Costa, P.C.; Jordan, S.; Ostrander, A. Quantum algorithm for simulating the wave equation. *Phys. Rev. A* **2019**, *99*, 012323. [CrossRef]
12. Jin, S.; Liu, N.; Yu, Y. Quantum simulation of partial differential equations via Schrodingerisation: Technical details. *arXiv* **2022**, arXiv:2212.14703.
13. Leyton, S.K.; Osborne, T.J. A quantum algorithm to solve nonlinear differential equations. *arXiv* **2008**, arXiv:0812.4423.

14. Liu, J.P.; Kolden, H.Ø.; Krovi, H.K.; Loureiro, N.F.; Trivisa, K.; Childs, A.M. Efficient quantum algorithm for dissipative nonlinear differential equations. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e2026805118. [CrossRef] [PubMed]
15. Itani, W.; Succi, S. Analysis of Carleman linearization of lattice Boltzmann. *Fluids* **2022**, *7*, 24. [CrossRef]
16. An, D.; Fang, D.; Jordan, S.; Liu, J.P.; Low, G.H.; Wang, J. Efficient quantum algorithm for nonlinear reaction-diffusion equations and energy estimation. *arXiv* **2022**, arXiv:2205.01141.
17. Krovi, H. Improved quantum algorithms for linear and nonlinear differential equations. *Quantum* **2023**, *7*, 913. [CrossRef]
18. Joseph, I. Koopman–von Neumann approach to quantum simulation of nonlinear classical dynamics. *Phys. Rev. Res.* **2020**, *2*, 043102. [CrossRef]
19. Engel, A.; Smith, G.; Parker, S.E. Linear embedding of nonlinear dynamical systems and prospects for efficient quantum algorithms. *Phys. Plasmas* **2021**, *28*, 062305 . [CrossRef]
20. Lin, Y.T.; Lowrie, R.B.; Aslangil, D.; Subaşı, Y.; Sornborger, A.T. Koopman von Neumann mechanics and the Koopman representation: A perspective on solving nonlinear dynamical systems with quantum computers. *arXiv* **2022**, arXiv:2202.02188.
21. Jin, S.; Liu, N. Quantum algorithms for computing observables of nonlinear partial differential equations. *arXiv* **2022**, arXiv:2202.07834.
22. Mauroy, A.; Susuki, Y.; Mezić, I. *Koopman Operator in Systems and Control*; Springer: Berlin/Heidelberg, Germany, 2020.
23. Schmid, P.J. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* **2010**, *656*, 5–28. [CrossRef]
24. Rowley, C.W.; Mezić, I.; Bagheri, S.; Schlatter, P.; Henningson, D.S. Spectral analysis of nonlinear flows. *J. Fluid Mech.* **2009**, *641*, 115–127. [CrossRef]
25. Askham, T.; Kutz, J.N. Variable projection methods for an optimized dynamic mode decomposition. *SIAM J. Appl. Dyn. Syst.* **2018**, *17*, 380–416. [CrossRef]
26. Williams, M.O.; Hemati, M.S.; Dawson, S.T.; Kevrekidis, I.G.; Rowley, C.W. Extending data-driven Koopman analysis to actuated systems. *IFAC-PapersOnLine* **2016**, *49*, 704–709. [CrossRef]
27. Manojlović, I.; Fonoberova, M.; Mohr, R.; Andrejčuk, A.; Drmač, Z.; Kevrekidis, Y.; Mezić, I. Applications of Koopman mode analysis to neural networks. *arXiv* **2020**, arXiv:2006.11765.
28. Lusch, B.; Kutz, J.N.; Brunton, S.L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.* **2018**, *9*, 4950. [CrossRef]
29. Brunton, S.L.; Budišić, M.; Kaiser, E.; Kutz, J.N. Modern Koopman theory for dynamical systems. *arXiv* **2021**, arXiv:2102.12086.
30. Li, B.; Yu, Y.; Yang, X. The Sparse-Grid-Based Adaptive Spectral Koopman Method. *arXiv* **2022**, arXiv:2206.09955.
31. Li, B.; Ma, Y.; Kutz, J.N.; Yang, X. The adaptive spectral koopman method for dynamical systems. *SIAM J. Appl. Dyn. Syst.* **2023**, *22*, 1523–1551. [CrossRef]
32. Trefethen, L.N. *Spectral Methods in MATLAB*; SIAM: Philadelphia, PA, USA, 2000.
33. Shen, J.; Tang, T.; Wang, L.L. *Spectral Methods: Algorithms, Analysis and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011; Volume 41.
34. Boyd, J.P. *Chebyshev and Fourier Spectral Methods*; Courier Corporation: North Chelmsford, MA, USA, 2001.
35. Wendland, H. Meshless Galerkin methods using radial basis functions. *Math. Comput.* **1999**, *68*, 1521–1531. [CrossRef]
36. Karniadakis, G.; Sherwin, S.J. *Spectral/hp Element Methods for Computational Fluid Dynamics*; Oxford University Press: New York, NY, USA, 2005.
37. Kowalski, K.; Steeb, W.H. *Nonlinear Dynamical Systems and Carleman Linearization*; World Scientific: Singapore, 1991.
38. Koopman, B.O. Hamiltonian systems and transformation in Hilbert space. *Proc. Natl. Acad. Sci. USA* **1931**, *17*, 315–318. [CrossRef]
39. Mezić, I. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dyn.* **2005**, *41*, 309–325. [CrossRef]
40. Bomze, I.M. Lotka-Volterra equation and replicator dynamics: A two-dimensional classification. *Biol. Cybern.* **1983**, *48*, 201–211. [CrossRef]
41. Orszag, S.A.; Bissonnette, L. Dynamical Properties of Truncated Wiener-Hermite Expansions. *Phys. Fluids* **1967**, *10*, 2603–2613. [CrossRef]
42. Forets, M.; Pouly, A. Explicit error bounds for Carleman linearization. *arXiv* **2017**, arXiv:1711.02552.
43. Amini, A.; Sun, Q.; Motee, N. Error bounds for Carleman linearization of general nonlinear systems. In Proceedings of the 2021 Conference on Control and its Applications, Virtual, 19–21 July 2021 ; pp. 1–8.