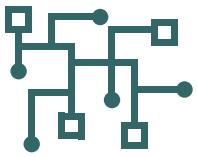


ISE

Industrial and Systems Engineering

The Adaptive Spectral Koopman Method for Dynamical Systems

BIAN LI[1], YI-AN MA[2], J. NATHAN KUTZ[3], AND XIU YANG[1]

[1]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

[2]Halıcıoğlu Data Science Institute and Department of Computer Science and Engineering, University of California San Diego, San Diego, CA, USA

[3]Department of Applied Mathematics, University of Washington, Seattle, WA, USA

ISE Technical Report 23T-005

LEHIGH UNIVERSITY

The Adaptive Spectral Koopman Method for Dynamical Systems*

Bian Li†, Yi-An Ma‡, J. Nathan Kutz§, and Xiu Yang†

Abstract. Dynamical systems have a wide range of applications in mechanics, electrical engineering, chemistry, and so on. In this work, we propose the adaptive spectral Koopman (ASK) method to solve nonlinear autonomous dynamical systems. This novel numerical method leverages the spectral-collocation (i.e., pseudo-spectral) method and properties of the Koopman operator to obtain the solution of a dynamical system. Specifically, this solution is represented as a linear combination of the multiplication of Koopman operator's eigenfunctions and eigenvalues, and these eigenpairs are approximated using the spectral method. Unlike conventional time evolution algorithms such as Euler's scheme and the Runge-Kutta scheme, ASK is mesh-free, and hence is more flexible when evaluating the solution. Numerical experiments demonstrate high accuracy of ASK for solving one-, two- and three-dimensional dynamical systems.

Key words. dynamical systems, Koopman operator, spectral-collocation method

AMS subject classifications. 65L05, 65L15, 58C40

1. Introduction.

The Koopman operator, introduced in 1931 by B. O. Koopman [11], is an infinite-dimensional *linear* operator that describes the evolution of a set of observables rather than the system state itself. The Koopman operator approach to *nonlinear* dynamical systems has attracted considerable attention recently, as it provides a rigorous method for globally linearizing the system dynamics. Specifically, because it is a linear operator, one can define its eigenvalues, eigenfunctions, and modes, and use them to represent dynamically interpretable low-dimensional embeddings of high-dimensional state spaces, which helps to understand the behavior of the underlying system and construct solutions through linear superposition [4]. In this procedure, the system dynamics is typically decomposed into linearly independent Koopman modes even if the system is nonlinear. In particular, as pointed out in [18, 12, 19], if the dynamics is ergodic but non-chaotic, the spectrum of the Koopman operator in properly defined spaces does not contain continuous spectra, and the observable of the system can be represented as a linear combination of eigenfunctions associated with discrete eigenvalues of the Koopman operator.

The Koopman operator provides powerful analytic tools to understand behaviors of dy-

*Submitted to the editors on March 30, 2022.

Funding: XY was supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research (ASCR) as part of Multifaceted Mathematics for Rare, Extreme Events in Complex Energy and Environment Systems (MACSER). BL was partially supported by Los Alamos National Laboratory. YM was supported in part by the National Science Foundation Grants NSF-SCALE MoDL(2134209) and NSF-CCF-2112665, and the Facebook research award. The work of JNK was supported in part by the US National Science Foundation (NSF) AI Institute for Dynamical Systems (dynamicsai.org), grant 2112085.

†Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA (bil215@lehigh.edu, xiy518@lehigh.edu)

‡Halıcıoğlu Data Science Institute & Department of Computer Science and Engineering, University of California San Diego, San Diego, CA (yianma@ucsd.edu)

§Department of Applied Mathematics, University of Washington, Seattle, WA (kutz@uw.edu)

namical systems. For example, dynamical evolution of a finite-dimensional system described by ordinary differential equations (ODEs) can be studied by conducting Koopman mode analysis. Such analysis starts with a choice of a set of linearly independent observables, and the Koopman operator is then analyzed through its action on the subspace spanned by the chosen observables [17]. Moreover, it is also shown that the Koopman operator approach can be formally generalized to infinite-dimensional dynamical systems described by partial differential equations (PDEs), providing new perspectives on the analysis and control of these nonlinear spatiotemporal dynamics [35, 20, 22, 19]. In addition, ergodic quotients and eigenquotients allow the Koopman operator to be used for the extraction and analysis of *invariant* and *periodic* structures in the state space [5]. Moreover, Mezić provided a Hilbert space setting for spectral analysis of *disspative dynamical systems*, and proved that the spectrum of the Koopman operator on these spaces is the closure of the product of the "on-attractor" and "off-attractor" spectra [18].

On the computational side, most existing numerical schemes motivated by the Koopman operator are categorized as data-driven methods, as they use spatiotemporal data to approximate a few of the leading Koopman eigenvalues, eigenfunctions, and modes. In particular, the emerging computational method *dynamics mode decomposition* (DMD) [24, 25, 30, 23, 14, 20, 1] as well as its variant such as extended DMD (EDMD) [34] uses snapshots of a dynamical system to extract temporal features as well as correlated spatial activity via matrix decomposition techniques. DMD and EDMD produce results for any appropriately formatted set of data, but connecting these outputs to the Koopman operator requires additional knowledge about the nature of the underlying system in that the system should be autonomous. Later, a modified EDMD [33] was proposed to compensate for the effects of system actuation when it is used to explore state space during the data collection, reestablishing the connection between EDMD and the Koopman operator in this more general class of data sets. A review of many of the DMD variants for approximating the Koopman operator can be found in Brunton et al [4].

Our aim in this paper is to provide a numerical method based on the spectral-collocation method (i.e., the pseudospectral method) to implement the Koopman-operator approach to solving nonlinear ordinary differential equations (ODEs). Unlike the data-driven methods, this approach is on the other end of the "spectrum" of numerical methods, as it is based on the classical spectral method [8, 29]. The main idea is to approximate eigenvalues, eigenfunctions, and modes of the Koopman operator based on its discretized form. Specifically, this method uses the differentiation matrix in spectral method to approximate the generator of the Koopman operator, and then conducts eigendecomposition numerically to obtain eigenvalues and eigenvectors that approximate Koopman operator's eigenvalues and eigenfunctions, respectively. Here, each element of an eigenvector is the approximation of the associated eigenfunction evaluated at a collocation point. The modes are approximated using the computed eigenvalues, eigenvectors, and the initial state (or observable). This work focuses on autonomous systems, and it would serve as a starting point for a new framework of numerical methods for dynamical systems.

The paper is organized as follows. Background topics are introduced in section 2. Then, the adaptive spectral Koopman method is discussed in detail in section 3. We present the experimental results in section 4, and the discussion and conclusions follow in section 5.

2. Background.

2.1. Koopman operator. Borrowing notions from [13], we consider an autonomous system described by the ordinary differential equations

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathbf{f}(\boldsymbol{x}), \tag{2.1}$$

where the state $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^\top$ belongs to a d-dimensional smooth manifold $\mathscr{M}$, and the dynamics $\mathbf{f} : \mathscr{M} \to \mathscr{M}$ does not explicitly depend on time t. Here, $\mathbf{f}$ is a possibly nonlinear vector-valued smooth function, of the same dimension as $\boldsymbol{x}$. In many studies, we are concerned with the behavior of observables on the state space. To this end, we define an observable to be a scalar function $g : \mathscr{M} \to \mathbb{R}$, where g is an element of some function space $\mathcal{G}$ (e.g., $\mathcal{G} = L^2(\mathscr{M})$ as in [17]). The flow map $\mathbf{F}_t : \mathscr{M} \to \mathscr{M}$ induced by the dynamical system (2.1) depicts the evolution of the system as

$$\boldsymbol{x}(t_0 + t) = \mathbf{F}_t(\boldsymbol{x}(t_0)) = \boldsymbol{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\boldsymbol{x}(s))\, ds. \tag{2.2}$$

Now we define the Koopman operator for continuous-time dynamical systems as follows [18]:

Definition 2.1. *Consider a family of operators $\{\mathcal{K}_t\}_{t\geq 0}$ acting on the space of observables so that*

$$\mathcal{K}_t g(\boldsymbol{x}_0) = g(\mathbf{F}_t(\boldsymbol{x}_0)),$$

where $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$. We call the family of operators $\mathcal{K}_t$ indexed by time t the Koopman operators of the continuous-time system (2.1).

By definition, $\mathcal{K}_t$ is a *linear* operator acting on the function space $\mathcal{G}$ for each fixed t. Moreover, $\{\mathcal{K}_t\}$ form a semi-group.

2.2. Infinitesimal generator. The Koopman spectral theory [17, 24] reveals properties that enable the Koopman operator to convert nonlinear finite-dimensional dynamics into linear infinite-dimensional dynamics. A key component in such spectral analysis is the initesimal generator (or generator for brievity) of the Koopman operator. Specifically, the generator of the Koopman operator $\mathcal{K}_t$, denoted as $\mathcal{K}$, is given by

$$\mathcal{K} g = \lim_{t\to 0} \frac{\mathcal{K}_t g - g}{t}. \tag{2.3}$$

For any smooth function g, (2.3) implies that

$$\mathcal{K} g(\boldsymbol{x}) = \frac{\mathrm{d}g(\boldsymbol{x})}{\mathrm{d}t} = \nabla g(\boldsymbol{x}) \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}. \tag{2.4}$$

Denoting φ an eigenfunction of $\mathcal{K}$ and λ the eigenvalue associated with φ, we have

$$\mathcal{K}\varphi(\boldsymbol{x}) = \lambda\varphi(\boldsymbol{x}). \tag{2.5}$$

Thus,

$$\lambda\varphi(\boldsymbol{x}) = \mathcal{K}\varphi(\boldsymbol{x}) = \frac{\mathrm{d}\varphi(\boldsymbol{x})}{\mathrm{d}t}. \tag{2.6}$$

This implies that $\varphi(\boldsymbol{x}(t_0+t)) = e^{\lambda t}\varphi(\boldsymbol{x}(t_0))$, i.e.,

$$\mathcal{K}_t\varphi(\boldsymbol{x}(t_0)) = e^{\lambda t}\varphi(\boldsymbol{x}(t_0)). \tag{2.7}$$

Therefore, φ is an eigenfunction of $\mathcal{K}_t$ associated with eigenvalue λ. Of note, following the conventional notation, the eigenpair for $\mathcal{K}_t$ is considered as (φ, λ) instead of $(\varphi, e^{\lambda t})$.

Now suppose g exists in the function space spanned by all the eigenfunctions φ_j (associated with eigenvalues λ_j) of $\mathcal{K}$, i.e., $g(\boldsymbol{x}) = \sum_j c_j\,\varphi_j(\boldsymbol{x})$, then

$$\mathcal{K}_t[g(\boldsymbol{x}(t_0))] = \mathcal{K}_t\left[\sum_j c_j\,\varphi_j(\boldsymbol{x}(t_0))\right] = \sum_j c_j\mathcal{K}_t[\varphi_j(\boldsymbol{x}(t_0))]. \tag{2.8}$$

Hence,

$$g(\boldsymbol{x}(t_0+t)) = \sum_j c_j\,\varphi_j(\boldsymbol{x}(t_0))e^{\lambda_j t}. \tag{2.9}$$

Similarly, if we choose a vector-valued observable $\boldsymbol{g}: \mathscr{M} \to \mathbb{R}^d$ with $\boldsymbol{g} := (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_d(\boldsymbol{x}))^\top$, the system of observables becomes

$$\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x})}{\mathrm{d}t} = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix}\mathcal{K}g_1(\boldsymbol{x})\\ \mathcal{K}g_2(\boldsymbol{x})\\ \vdots\\ \mathcal{K}g_d(\boldsymbol{x})\end{bmatrix} = \sum_j \lambda_j\,\varphi_j(\boldsymbol{x})\boldsymbol{c}_j, \tag{2.10}$$

where $\boldsymbol{c}_j \in \mathbb{C}^d$ is called the jth Koopman mode with $\boldsymbol{c}_j := (c_j^1, c_j^2, \ldots, c_j^d)^\top$. In general, there is no universal guide for choosing observables as this choice is problem dependent. A good set of observables can lead to a system that is significantly easier to solve. An example from [3, 16] is illustrated in Appendix A.

We finalize the introduction of the Koopam operator with the following simple example. Consider the system $\frac{\mathrm{d}x}{\mathrm{d}t} = \mu x$ with $x, \mu \in \mathbb{R}$ and $\mu \neq 0$. Then, one can easily verify that $\varphi_n(x) := x^n$ is an eigenfunction of the Koopman operator associated with this dynamical system, and the corresponding eigenvalue is $\lambda_n = n\mu$ with $n \in \mathbb{N}^+$ (a similar example is presented in [6]). According to (2.9), by setting $g(x) = x$ and let $x(0) = x_0$, we have

$$x(t) = \sum_{j=1}^{\infty} c_j\,\varphi_j(x_0)e^{\lambda_j t} = \sum_{j=1}^{\infty} c_j x_0^j e^{\mu j t}.$$

Setting $t = 0$ gives $x_0 = x(0) = \sum_{j=1}^{\infty} c_j x_0^j$, which indicates $c_1 = 1$ and $c_j = 0$ when $j \neq 1$. Therefore, we obtain the solution of the ODE as $x(t) = x_0 e^{\mu t}$.

3. Adaptive Spectral Koopman Method.

In this section, we introduce the adaptive spectral Koopman (ASK) method, which is a numerical method based on the Koopman operator and the spectral method to solve ODE systems. Before describing details of this method, we introduce the notations used in this algorithm. Let $\boldsymbol{x}(t)$ denote the solution of an ODE system with an initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$. Assume $t_0 = 0$ in (2.2), we consider solutions in time interval $[0, T]$ with $T > 0$. Letter n denotes the number of "check points" (see details in subsection 3.4). The radius of the neighborhood of $\boldsymbol{x}(t)$ is denoted by r while γ is a parameter that controls the update of the neighborhood.

3.1. Finite-dimensional approximation. Based on the preliminaries introduced in subsection 2.2, we aim to identify the following truncated approximation of (2.9)

$$g(\boldsymbol{x}(t)) \approx g_N(\boldsymbol{x}(t)) = \sum_{j=0}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}_0) e^{\tilde{\lambda}_j t}. \tag{3.1}$$

where φ_j^N are polynomial approximations of φ_j, $\tilde{\lambda}_j$ and $\tilde{c}_j$ approximate λ_j and c_j, respectively. Next, because $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathbf{f}(\boldsymbol{x})$, (2.4) and (2.6) indicate that for any eigenfunction φ,

$$\begin{aligned} \mathcal{K}\varphi = \mathbf{f}\cdot\nabla\varphi &= \left(f_1\frac{\partial\varphi}{\partial x_1} + f_2\frac{\partial\varphi}{\partial x_2} + \ldots + f_d\frac{\partial\varphi}{\partial x_d}\right) \\ &= \left(f_1\frac{\partial}{\partial x_1} + f_2\frac{\partial}{\partial x_2} + \ldots + f_d\frac{\partial}{\partial x_d}\right)(\varphi). \end{aligned}$$

Thus,

$$\mathcal{K} = f_1\frac{\partial}{\partial x_1} + f_2\frac{\partial}{\partial x_2} + \ldots + f_d\frac{\partial}{\partial x_d}. \tag{3.2}$$

Here, we consider the case with $d \le 3$, and adopt the approaches in the spectral-collocation method. Specifically, our algorithm uses Gauss-Lobatto points for the interpolation of φ and approximates (partial) derivatives with differentiation matrices (see e.g., [10, 26, 9]) in (3.2). Consequently, the first step is to discretize $\mathcal{K}$.

(1) When $d = 1$. Let $\{\xi_i\}_{i=0}^N$ be the Gauss-Lobatto points and the polynomial interpolation of $\varphi(\boldsymbol{x})$ is

$$\varphi(x) \approx \varphi^N(x) := \sum_{i=0}^{N} \varphi^N(\xi_i) P_i(x),$$

where the basis functions P_j are Lagrange polynomials satisfying $P_j(\xi_i) = \delta_{ij}$ and δ_{ij} is the Kronecker delta function. Namely, $\varphi^N(x)$ is the projection of $\varphi(x)$ on the space $\mathrm{span}\{P_j(x)\}_{j=0}^N$. Let $\boldsymbol{\varphi^N} = [\varphi^N(\xi_0), \varphi^N(\xi_1), \ldots, \varphi^N(\xi_N)]^\top$, we have

$$\mathcal{K}\boldsymbol{\varphi^N} = \mathrm{diag}(\mathbf{f}(\xi_0), \mathbf{f}(\xi_1), \ldots, \mathbf{f}(\xi_N))\mathbf{D}\boldsymbol{\varphi^N} := \mathbf{K}\boldsymbol{\varphi^N}, \tag{3.3}$$

where $\mathbf{D}$ is the differentiation matrix associated with $\{\xi_i\}_{i=0}^N$ and $\mathbf{K}$ is an $(N+1)\times(N+1)$ matrix. Here, we abuse the notation to let $\mathcal{K}\boldsymbol{\varphi^N} = [\mathcal{K}\varphi^N(\xi_0), \mathcal{K}\varphi^N(\xi_1), \ldots, \mathcal{K}\varphi^N(\xi_N)]^\top$, and similar notations are used in the following $d = 2, 3$ cases.

(2) When $d = 2$. Let $\{\xi_i\}_{i=0}^N$ and $\{\eta_j\}_{j=0}^N$ be the Gauss-Lobatto points of x_1 and x_2, respectively. Every eigenfunction φ is now a bivariate function, whose polynomial interpolation φ^N is

$$\varphi(x_1, x_2) \approx \varphi^N(x_1, x_2) := \sum_{i=0}^{N}\sum_{j=0}^{N} \varphi^N(\xi_i, \eta_j) P_i(x_1) P_j(x_2).$$

Hence, we define a matrix $\boldsymbol{\Phi}^N$ as

$$\boldsymbol{\Phi}^N = \begin{bmatrix} \varphi^N(\xi_0,\eta_0) & \varphi^N(\xi_0,\eta_1) & \cdots & \varphi^N(\xi_0,\eta_N) \\ \varphi^N(\xi_1,\eta_0) & \varphi^N(\xi_1,\eta_1) & \cdots & \varphi^N(\xi_1,\eta_N) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi^N(\xi_N,\eta_0) & \varphi^N(\xi_N,\eta_1) & \cdots & \varphi^N(\xi_N,\eta_N) \end{bmatrix}.$$

Let $\mathbf{D}_1$ and $\mathbf{D}_2$ be the differentiation matrices for x_1 and x_2, respectively, and $\mathbf{F}_1$ and $\mathbf{F}_2$ be the matrices of f_1 and f_2 evaluated at (ξ_i, η_j). Also, we denote $\mathcal{K}\mathbf{\Phi}^N$ the matrix with elements $(\mathcal{K}\mathbf{\Phi}^N)_{ij} = \mathcal{K}\mathbf{\Phi}^N(\xi_i, \eta_j)$. Then, $\mathcal{K}\mathbf{\Phi}^N$ can be computed as

$$\mathcal{K}\mathbf{\Phi}^N = \mathbf{F}_1 \odot \left(\mathbf{D}_1\mathbf{\Phi}^N\right) + \mathbf{F}_2 \odot \left(\mathbf{\Phi}^N\mathbf{D}_2^\top\right),$$

where $\odot$ denotes the Hadamard product. In the computation, we vectorize $\mathbf{\Phi}^N$ (along columns) to obtain

$$\begin{aligned}\mathcal{K}\,\mathrm{vec}(\mathbf{\Phi}^N) &= \mathrm{vec}(\mathbf{F}_1) \odot \Big((\mathbf{I} \otimes \mathbf{D}_1)\,\mathrm{vec}(\mathbf{\Phi}^N)\Big) + \mathrm{vec}(\mathbf{F}_2) \odot \Big((\mathbf{D}_2 \otimes \mathbf{I})\,\mathrm{vec}(\mathbf{\Phi}^N)\Big)\\ &= \Big[\,\mathrm{diag}\big(\,\mathrm{vec}(\mathbf{F}_1)\big)(\mathbf{I} \otimes \mathbf{D}_1) + \mathrm{diag}\big(\,\mathrm{vec}(\mathbf{F}_2)\big)(\mathbf{D}_2 \otimes \mathbf{I})\Big]\Big(\,\mathrm{vec}(\mathbf{\Phi}^N)\Big)\\ &:= \mathbf{K}\,\mathrm{vec}(\mathbf{\Phi}^N),\end{aligned}$$

where $\otimes$ denotes the Kronecker product, $\mathbf{I}$ is the identity matrix, and $\mathbf{K}$ is an $(N+1)^2 \times (N+1)^2$ matrix.

(3) When $d = 3$. Let $\{\xi_i\}_{i=0}^N$, $\{\eta_j\}_{j=0}^N$, and $\{\zeta_k\}_{k=0}^N$ be the Gauss-Lobatto points of x_1, x_2, and x_3, respectively. The collocation points are then (ξ_i, η_j, ζ_k). In this case, φ is approximated as

$$\varphi(x_1, x_2, x_3) \approx \varphi^N(x_1, x_2, x_3) := \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N} \varphi^N(\xi_i, \eta_j, \zeta_k) P_i(x_1) P_j(x_2) P_k(x_3).$$

Hence, the values of φ^N at the collocation points can be represented by a tensor $\mathbf{\Phi}^N$ whose frontal slices are written as

$$\mathbf{\Phi}^N(:,:,k) = \begin{bmatrix} \varphi^N(\xi_0, \eta_0, \zeta_k) & \varphi^N(\xi_0, \eta_1, \zeta_k) & \cdots & \varphi^N(\xi_0, \eta_N, \zeta_k) \\ \varphi^N(\xi_1, \eta_0, \zeta_k) & \varphi^N(\xi_1, \eta_1, \zeta_k) & \cdots & \varphi^N(\xi_1, \eta_N, \zeta_k) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi^N(\xi_N, \eta_0, \zeta_k) & \varphi^N(\xi_N, \eta_1, \zeta_k) & \cdots & \varphi^N(\xi_N, \eta_N, \zeta_k) \end{bmatrix}.$$

With the n-mode multiplication in tensor algebra, we arrive at a compact representation of the approximation,

$$\mathcal{K}\mathbf{\Phi}^N = \mathbf{F}_1 \odot \Big(\mathbf{\Phi}^N \times_1 \mathbf{D}_1\Big) + \mathbf{F}_2 \odot \Big(\mathbf{\Phi}^N \times_2 D_2\Big) + \mathbf{F}_3 \odot \Big(\mathbf{\Phi}^N \times_3 \mathbf{D}_3\Big),$$

where $\times_p$ denotes the mode-p tensor-matrix multiplication. Here, $\mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3$ are the differentiation matrices, and $\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3$ denote the tensors resulting from f_1, f_2, f_3 evaluated at (ξ_i, η_j, ζ_k). Following the same idea of vectorization, we rewrite the

tensor representation as

$$\begin{aligned}\mathcal{K}\,\mathrm{vec}(\mathbf{\Phi}^N) &= \mathrm{vec}(\mathbf{F}_1)\odot\Big((\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{D}_1)\,\mathrm{vec}(\mathbf{\Phi}^N)\Big)\\ &+\mathrm{vec}(\mathbf{F}_2)\odot\Big((\mathbf{I}\otimes\mathbf{D}_2\otimes\mathbf{I})\,\mathrm{vec}(\mathbf{\Phi}^N)\Big)\\ &+\mathrm{vec}(\mathbf{F}_3)\odot\Big((\mathbf{D}_3\otimes\mathbf{I}\otimes\mathbf{I})\,\mathrm{vec}(\mathbf{\Phi}^N)\Big)\\ &=\Big[\,\mathrm{diag}\big(\,\mathrm{vec}(\mathbf{F}_1)\big)(\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{D}_1)\\ &+\mathrm{diag}\big(\,\mathrm{vec}(\mathbf{F}_2)\big)(\mathbf{I}\otimes\mathbf{D}_2\otimes\mathbf{I})\\ &+\mathrm{diag}\big(\,\mathrm{vec}(\mathbf{F}_3)\big)(\mathbf{D}_3\otimes\mathbf{I}\otimes\mathbf{I})\Big]\Big(\,\mathrm{vec}(\mathbf{\Phi}^N)\Big)\\ &:=\mathbf{K}\,\mathrm{vec}(\mathbf{\Phi}^N).\end{aligned}$$

where $\mathbf{K}$ is an $(N+1)^3\times(N+1)^3$ matrix.

In all these cases, the discretized generator $\mathcal{K}$ can be represented as a matrix $\mathbf{K}$. For $d=2$ and $d=3$, the total number of eigenfuncitons used in (3.1) is $(N+1)^2$ and $(N+1)^3$, respectively, instead of $(N+1)$. For brevity, g_N is still used to denote the approximated observable for different d. The derivation of higher dimensional systems amounts to further extensions of the three-dimensional case by the Kronecker product.

3.2. Eigen-decomposition. Now the eigenvalue problem of the Koopman operator in (2.5) is discretized as the eigenvalue problem of matrix $\mathbf{K}$, i.e., $\mathbf{K}\boldsymbol{v}=\tilde{\lambda}\boldsymbol{v}$, where $\tilde{\lambda}\in\mathbb{C}$ and $\boldsymbol{v}$ is a complex vector. The vector $\boldsymbol{v}$ is an approximation of $\mathcal{K}$'s eigenfunction φ evaluated at the collocation points and $\tilde{\lambda}$ is the approximation of the associated eigenvalue of $\mathcal{K}$. The matrix form of the eigenvalue problem is

$$\mathbf{K}\mathbf{V}=\mathbf{V}\mathbf{\Lambda}, \tag{3.4}$$

where $\mathbf{V}$ consists of columns $\boldsymbol{v}_j$ and the diagonal elements of $\mathbf{\Lambda}$ are $\tilde{\lambda}_j$. By construction, for $d=1$, $(\boldsymbol{v}_j)_i=\varphi_j^N(\xi_i)\approx\varphi_j(\xi_i)$, and for $d=2$ or 3, $\boldsymbol{v}_j=\mathrm{vec}(\mathbf{\Phi}_j^N)$, where $\mathbf{\Phi}_j^N$ approximates the values of eigenfunction φ_j at the collocation points. Of note, the collocation points in multi-dimensional cases are constructed by the tensor product of one-dimensional collocation points, but we have not specified how to obtain such points, the details of which are given in subsection 3.3. Also, we emphasize that these collocation points are related to $\boldsymbol{x}$ instead of t. In other words, ASK discretizes $\varphi(\boldsymbol{x})$ in space instead of discretizing $\boldsymbol{x}(t)$ in time, which is different from conventional spectral methods for ODEs.

3.3. Constructing the solution. Let us first consider $d=1$. By the eigen-decomposition, one can access values of eigenfunctions at the Gauss-Lobatto points $\mathbf{\Xi}:=\{\xi_i\}_{i=0}^N$, where $\xi_0<\xi_1<\ldots<\xi_N$. Therefore, $\varphi(\boldsymbol{x}_0)$ can be approximated when $\xi_0\le\boldsymbol{x}_0\le\xi_N$. To avoid polynomial interpolation, ASK uses an even number for N and sets $\xi_{N/2}=\boldsymbol{x}_0$. Based on this setting, we consider a neighborhood of $\boldsymbol{x}_0$ with radius r, i.e., $[\boldsymbol{x}_0-r,\boldsymbol{x}_0+r]$, where r is tunable. Gauss-Lobatto points are then generated such that $\boldsymbol{x}_0-r=\xi_0<\xi_2<\ldots<\xi_{N/2}=\boldsymbol{x}_0<\ldots<\xi_N=\boldsymbol{x}_0+r$. Thus, g_N is constructed as

$$g_N(\boldsymbol{x}(t))=\sum_{j=0}^{N}\tilde{c}_j\varphi_j^N(\boldsymbol{x}_0)e^{\tilde{\lambda}_j t}=\sum_{j=0}^{N}\tilde{c}_j\varphi_j^N(\xi_{N/2})e^{\tilde{\lambda}_j t}=\sum_{j=0}^{N}\tilde{c}_j(\boldsymbol{v}_j)_{N/2}e^{\tilde{\lambda}_j t}, \tag{3.5}$$

where $\boldsymbol{v}_j$ are eigenvectors of matrix $\mathbf{K}$ computed in subsection 3.1.

To approximate Koopman modes c_j, we set $t = 0$ in (3.5), which yields

$$g(\boldsymbol{x}_0) \approx g_N(\boldsymbol{x}_0) = \sum_{j=0}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}_0),$$

which holds for different initial state $\boldsymbol{x}_0$, e.g.,

$$g(\xi_i) \approx g_N(\xi_i) = \sum_{j=0}^{N} \tilde{c}_j \varphi_j^N(\xi_i), \quad i = 0, \ldots, N,$$

where ξ_i are the aforementioned Gauss-Lobatto points. Thus, we can obtain $\tilde{c}_j$ by solving a linear system $\mathbf{V}\boldsymbol{c} = g(\boldsymbol{\Xi})$, where $\mathbf{V}$ is defined in (3.4), $g(\boldsymbol{\Xi}) = (g(\xi_0), \ldots, g(\xi_N))^\top$ and $\boldsymbol{c} = (\tilde{c}_0, \ldots, \tilde{c}_N)^\top$. As an example, if $g(\boldsymbol{x}) \coloneqq \boldsymbol{x}$, then $g(\boldsymbol{\Xi}) = (\xi_0, \ldots, \xi_N)^\top$.

For $d = 2$, we consider the neighborhood of $\boldsymbol{x}_0 = (x_0^1, x_0^2)^\top$ as $[x_0^1 - r, x_0^1 + r] \times [x_0^2 - r, x_0^2 + r]$. Similarly, for $d = 3$, the neighborhood is $[x_0^1 - r, x_0^1 + r] \times [x_0^2 - r, x_0^2 + r] \times [x_0^3 - r, x_0^3 + r]$, where $\boldsymbol{x}_0 = (x_0^1, x_0^2, x_0^3)^\top$. We then generate $(N+1)$ Gauss-Lobatto points in each direction and use the tensor product rule to construct multi-dimensional collocation points. In practice, one can use standard Gauss-Lobatto points in the spectral method such as Legendre-Gauss-Lobatto and Chebyshev-Gauss-Lobatto points. Now the set of all collocation points is $\boldsymbol{\Xi} = \{(\xi_i, \eta_j)\}_{i,j=0}^N$ for $d = 2$ and $\boldsymbol{\Xi} = \{(\xi_i, \eta_j, \zeta_k)\}_{i,j=0}^N$ for $d = 3$. Of note, the isotropic set up is applied here for demonstration purpose, i.e., we use a fixed r in each direction and admit the same number of Gauss-Lobatto points in each dimension. However, this is not necessarily the optimal choice, and one can use different r and different numbers of Gauss-Lobatto points in different directions.

Next, since we vectorize matrix (or tensor) $\boldsymbol{\Phi}^N$ column by column (or slice by slice) as shown in subsection 3.1, $\varphi_j(\boldsymbol{x}_0)$ is again approximated by the "middle" element of vector $\mathrm{vec}(\boldsymbol{\Phi}_j)$, which leads to

$$\boldsymbol{g}_N(\boldsymbol{x}(t)) = \begin{cases} \sum_{j=0}^{(N+1)^2-1} \boldsymbol{c}_j \, (\boldsymbol{v}_j)_{[(N+1)^2-1]/2} \, e^{\tilde{\lambda}_j t}, & d = 2; \\ \sum_{j=0}^{(N+1)^3-1} \boldsymbol{c}_j \, (\boldsymbol{v}_j)_{[(N+1)^3-1]/2} \, e^{\tilde{\lambda}_j t}, & d = 3. \end{cases} \tag{3.6}$$

Here, each element of the modes $\boldsymbol{c}_j = (\tilde{c}_j^1, \cdots, \tilde{c}_j^d)^\top$ corresponds to a component of $\boldsymbol{g}$, and it is computed in the same manner as in the $d = 1$ case. For example, for $d = 2$, i.e., $\boldsymbol{g}(\boldsymbol{x}) = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}))^\top$ (correspondingly, $\boldsymbol{g}_N(\boldsymbol{x}) = (g_N^1(\boldsymbol{x}), g_N^2(\boldsymbol{x}))^\top$), we have $g_N^1(\boldsymbol{x}(t)) = \sum_{j=0}^{(N+1)^2-1} \tilde{c}_j^1 \, \mathrm{vec}(\boldsymbol{\Phi}_j)_{[(N+1)^2-1]/2} e^{\tilde{\lambda}_j t}$. Consider matrix $g_1(\boldsymbol{\Xi})$ whose elements are $(g_1(\boldsymbol{\Xi}))_{ij} = g_1(\xi_i, \eta_j)$. The modes $\boldsymbol{c}^1 = (\tilde{c}_0^1, \ldots, \tilde{c}_N^1)^\top$ are obtained by solving a linear system $\mathbf{V}\boldsymbol{c}^1 = \mathrm{vec}(g_1(\boldsymbol{\Xi}))$. Similarly, we can compute the modes for $g_N^2(\boldsymbol{x})$. In practice, our algorithm solves the linear system $\mathbf{V}\mathbf{C} = \boldsymbol{g}(\boldsymbol{\Xi})$, where $\mathbf{C} = (\boldsymbol{c}^1, \boldsymbol{c}^2)$ and $\boldsymbol{g}(\boldsymbol{\Xi}) = (\mathrm{vec}(g_1(\boldsymbol{\Xi})), \mathrm{vec}(g_l(\boldsymbol{\Xi})))$. The modes for $d = 3$ are computed in the same manner. In addition, a pseudocode is presented in Appendix B to illustrate how the solution is constructed.

3.4. Adaptivity. Since we apply a finite-dimensional approximation of the Koopman operator and exploit the Lagrange interpolation to approximate the eigenfunctions, the accuracy of the solution may decay as time evolves, especially for highly nonlinear systems. To further improve the accuracy, we propose an adaptive approach to update $\mathbf{V}, \mathbf{\Lambda}$ and $\boldsymbol{c}_j$. The main idea is to identify the time to repeat the procedure described in subsection 3.1– subsection 3.3. To this end, we set check points $0 < \tau_1 < \tau_2 < \ldots < \tau_n < T$ to examine the "validity" of the neighborhood of $\boldsymbol{x}(\tau_k)$. Specifically, the component of $\boldsymbol{x}(\tau_k) = (x_1(\tau_k), \ldots, x_d(\tau_k))^\top$ is acceptable if $x_i(\tau_k) \in R_i$ where

$$R_i := [L_i + \gamma r_i, U_i - \gamma r_i]. \tag{3.7}$$

Here, L_i and U_i are the lower and upper bounds, r_i is the radius in the ith direction, and $\gamma \in (0, 1]$ is a tunable parameter. Recall that the isotropic setup is used in this work, thus $r_i \equiv r$. In the initial step, $L_i := x_0^i - r_i$ and $U_i := x_0^i + r_i$, i.e., $\gamma = 1$. In practice, one can fix $\gamma = 1$ (or other real number in $(0, 1]$) and tune r_i only. Hereby, we keep both γ and r_i for future extension to anisotropic design and more advanced adaptivity criterion.

If $x_i(\tau_k) \in R_i$ for all i, then $R_1 \times \ldots \times R_d$ is a valid neighborhood of $\boldsymbol{x}(\tau_k)$. Otherwise, we update all L_i, U_i and reconstruct $\varphi_j^N, \tilde{\lambda}_j, \tilde{c}_j$ to obtain $\boldsymbol{x}(t)$ $(t > \tau_k)$ as follows:

1. Set $L_i = x_i(\tau_k) - r_i, U_i = x_i(\tau_k) + r_i$, $1 \le i \le d$.
2. Generate Gauss-Lobatto points and the differentiation matrix in each interval $[L_i^k, U_i^k]$. Repeat the procedure in subsection 3.1 to compute matrix $\mathbf{K}$.
3. Repeat the eigendecomposition in subsection 3.2 to update $\mathbf{V}$ and $\mathbf{\Lambda}$ in (3.4).
4. Compute coefficient $\boldsymbol{c}_j$ as in subsection 3.3 with the updated $\mathbf{V}$.
5. Construct solution $\boldsymbol{x}(t)$ by replacing $e^{\tilde{\lambda}_j t}$ with $e^{\tilde{\lambda}_j (t-\tau_k)}$ in (3.5) (or (3.6) for $d = 2, 3$).

Note that the modification of constructing the solution in step 5 is necessary because when an update is performed, we need to set $t_0 = \tau_k$ and $\boldsymbol{x}_0 = \boldsymbol{x}(t_0) = \boldsymbol{x}(\tau_k)$.

The parameter γ decides how often we update the neighborhood and reconstruct the solution. By construction, a larger γ demands updating the eigendecomposition more frequently. The extreme case $\gamma = 1$ enforces the update at every check point. In this work, we set $\tau_{k+1} - \tau_k \equiv \Delta\tau$. Notably, since the solution is discretized *in space* instead of in time as in conventional ODE solvers, the check points are different from time grids $0 < t_1 < t_2 < \ldots$ in those solvers. If we set $k = 0$, then no update is made, which indicates that the solution $\boldsymbol{x}(t)$ only relies on the eigendecomposition based on $\boldsymbol{x}_0$ (see the example pseudocode in Appendix Appendix B).

3.5. Properties of the algorithm. In this work $\boldsymbol{x}, \mathbf{f}, \boldsymbol{g}$ are real-valued functions. Now we show that the solutions obtained by ASK are real numbers, although $\mathbf{V}, \mathbf{\Lambda}, \boldsymbol{c}_j$ may contain complex values. We start with reiterating a well-known conclusion:

Lemma 3.1. *If a real matrix has complex eigenvalues, then they always occur in complex conjugate pairs. Furthermore, a complex conjugate pair of eigenvalues have a complex conjugate pair of associated eigenvectors.*

Proof. Suppose the matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ has an eigenpair $\boldsymbol{v}$ and λ such that $\mathbf{K}\boldsymbol{v} = \lambda\boldsymbol{v}$. Let $\bar{\cdot}$ operator denote the complex conjugate. Taking the complex conjugate of both sides of the equation, we have $\bar{\mathbf{K}}\bar{\boldsymbol{v}} = \bar{\lambda}\bar{\boldsymbol{v}}$. However, $\mathbf{K} = \bar{\mathbf{K}}$ since $\mathbf{K}$ has real entries. Thus, $\mathbf{K}\bar{\boldsymbol{v}} = \bar{\lambda}\bar{\boldsymbol{v}}$. The claim follows. ∎

Our main theorem is presented next:

Theorem 3.2. *ASK yields real-valued solutions for dynamical systems with real-valued* $\boldsymbol{x}, \mathbf{f}$ *and* $\boldsymbol{g}$.

Proof. We only need to consider the $d = 1$ case since the solution for high-dimensional cases are constructed in the same manner. Let $\boldsymbol{v}$ be a eigenvector, then it is a column of matrix $\mathbf{V}$ in (3.4). It is only necessary to consider the case where $\boldsymbol{v}$ is not a real-valued vector. According to Lemma 3.1, $\bar{\boldsymbol{v}}$ is also a column of $\mathbf{V}$. Let $\boldsymbol{u}$ be a row of $\mathbf{V}^{-1}$ such that $\boldsymbol{u} \cdot \boldsymbol{v} = 1$ and $\boldsymbol{u} \cdot \tilde{\boldsymbol{v}} = 0$, where $\tilde{\boldsymbol{v}}$ is any column of $\mathbf{V}$ other than $\boldsymbol{v}$. It is clear that $\bar{\boldsymbol{u}} \cdot \bar{\boldsymbol{v}} = 1$ and $\bar{\boldsymbol{u}} \cdot \tilde{\boldsymbol{v}} = 0$. Therefore, $\bar{\boldsymbol{u}}$ is also a row of $\mathbf{V}^{-1}$. Next, as shown in subsection 3.3, we compute the modes $\boldsymbol{c}$ as $\boldsymbol{c} = \mathbf{V}^{-1}\boldsymbol{g}(\boldsymbol{\Xi})$. Let c_m be the element of $\boldsymbol{c}$ such that $c_m = \boldsymbol{u} \cdot \boldsymbol{g}(\boldsymbol{\Xi})$, then $\bar{\boldsymbol{u}} \cdot \boldsymbol{g}(\boldsymbol{\Xi}) = \overline{\boldsymbol{u} \cdot \boldsymbol{g}(\boldsymbol{\Xi})} = \bar{c}_m$ is also an element of $\boldsymbol{c}$.

In the numerical solution, it suffices to consider $c_m \nu e^{\lambda t} + \bar{c}_m \bar{\nu} e^{\bar{\lambda} t}$, where $\nu \in \mathbb{C}$ denotes the middle element of the eigenvector $\boldsymbol{v}$. For convenience, we dentoe $\nu = A + Bi, \lambda = C + Di, c_m = E + Fi$. Here, $A, B, C, D, E, F \in \mathbb{R}$. Then,

$$
\begin{aligned}
c_m \nu e^{\lambda t} + \bar{c}_m \bar{\nu} e^{\bar{\lambda} t} &= (E + Fi)(A + Bi)e^{(C+Di)t} + (E - Fi)(A - Bi)e^{(C-Di)t} \\
&= (P + Qi)e^{(C+Di)t} + (P - Qi)e^{(C-Di)t} \\
&= \big(Pe^{Ct} + Qe^{Ct}i\big)e^{Dti} + \big(Pe^{Ct} - Qe^{Ct}i\big)e^{-Dti} \\
&= \big(Pe^{Ct} + Qe^{Ct}i\big)[\cos(Dt) + \sin(Dt)i] \\
&\quad + \big(Pe^{Ct} - Qe^{Ct}i\big)[\cos(Dt) - \sin(Dt)i] \\
&= 2Pe^{Ct}\cos(Dt) + 2\big(Qe^{Ct}i\big)\sin(Dt)i \\
&= 2Pe^{Ct}\cos(Dt) - 2Qe^{Ct}\sin(Dt) \in \mathbb{R},
\end{aligned}
$$

among which $P = AE - BF$ and $Q = AF + BE$. ∎

3.6. Algorithm summary. As a summary, subsection 3.1 to subsection 3.3 present a numerical scheme that solves an autonomous ODE system using the eigendecomposition and a linear system solver. Subsection 3.4 introduces a heuristic adaptivity criterion to repeat the aforementioned procedure at appropriate time points to further enhance the accuracy. We conclude the algorithm in Algorithm 3.1.

In the ASK scheme, the neighborhood for all components must be updated in the adaptivity step. This is because we set the current state of each component to be the midpoint of the corresponding neighborhood to avoid computing interpolation. Also, following the standard practice in the spectral method, we generate Gauss-Lobatto points ξ_i and the associated differentiation matrix $\mathbf{D}_i$ on $[-1, 1]$ first, and then scale them to $[L_i, U_i]$ as $\frac{U_i - L_i}{2}\big(\xi_i + 1\big) + L_i$ and $\frac{2\mathbf{D}_i}{U_i - L_i}$ to improve the computational efficiency. Moreover, it is worth emphasizing again that the isotropic setup (i.e., using the same number of Gauss-Lobatto points in each direction and fix $r_i \equiv r$) is not necessarily the optimal choice, and that the adaptivity in different directions may improve the efficiency of the algorithm. This is beyond the scope of this work and will be included in the future study.

4. Numerical Results. This section presents the performance of ASK on six nonlinear ODE systems including $d = 1, 2, 3$. In each example, we investigate the influence of num-

Algorithm 3.1 Adaptive spectral Koopman method

Require: $n, T, N, \boldsymbol{x}_0, r, \gamma$

1: Set check points at $0 = \tau_0 < \tau_1 < ... < \tau_n < T$.
2: Let $L_i = x_0^i - r_i, U_i = x_0^i + r_i$ and set neighborhood R_i as $R_i = [L_i + \gamma r_i, U_i - \gamma r_i]$ for $i = 1, 2, ..., d$, where $r_i = r$.
3: Generate Gauss-Lobatto points and differentiation matrix $\mathbf{D}_i$ in $[L_i, U_i]$ for $i = 1, 2, ..., d$. Construct collocation points $\boldsymbol{\Xi}$ for $d > 1$ using the tensor product rule. (For $d = 1$, $\boldsymbol{\Xi}$ is the set of the Gauss-Lobatto points.)
4: Construct matrix $\mathbf{K}$ using the formulas in subsection 3.1
5: Compute eigen-decomposition $\mathbf{KV} = \mathbf{V\Lambda}$
6: Solve linear system $\mathbf{VC} = \boldsymbol{g}(\boldsymbol{\Xi})$, where the lth column of matrix $\boldsymbol{g}(\boldsymbol{\Xi})$ consists of the lth component of all collocation points (see subsection 3.3).
7: **for** $k = 1, 2, 3, \ldots, n$ **do**
8: Let ν_j be the middle element of the jth column of $\mathbf{V}$. Construct solution at time τ_k as $\boldsymbol{x}(\tau_k) = \sum_j \mathbf{C}(j,:)\nu_j e^{\tilde{\lambda}_j(\tau_k - \tau_{k-1})}$, where $\mathbf{C}(j,:)$ is the jth row of $\mathbf{C}$.
9: **if** $(\boldsymbol{x}(\tau_k))_i \notin R_i$ for any i **then**
10: Reset $L_i = x_i(\tau_k) - r_i, U_i = x_i(\tau_k) + r_i$ and $R_i = [L_i + \gamma r_i, U_i - \gamma r_i]$.
11: Repeat steps $3 - 6$
12: **end if**
13: **end for**
14: **return** $\boldsymbol{x}(T) = \sum_j \mathbf{C}(j,:)\nu_j e^{\tilde{\lambda}_j(T - \tau_n)}$.

ber of Gauss-Lobatto points N, number of check points n, and the radius r on the accuracy. The reference solution is generated by Verner's ninth order Runge-Kutta (RK9) method [31] if a close-form solution is not available. We compare ASK with fourth order Runge-Kutta (RK4) method [7] since it is the most widely used method in the Runge-Kutta family. Throughout the numerical examples, ASK employs the Chebyshev-Gauss-Lobatto points. Additionally, we also tested Legendre-Gauss-Lobatto points but there was no significant difference. Moreover, in subsection 4.7 we compare the efficiency of ASK, RK4, and RK9 based on number of function evaluations for systems with closed-form solutions. Herein, the function refers to $\mathbf{f}$ in the ODE. (All the codes can be downloaded at https://github.com/Navarro33/Adaptive-Spectral-Koopman-Method)

4.1. Cosine model. The cosine model is a synthetic model invented for our demonstrative purposes. The governing ODE is written as

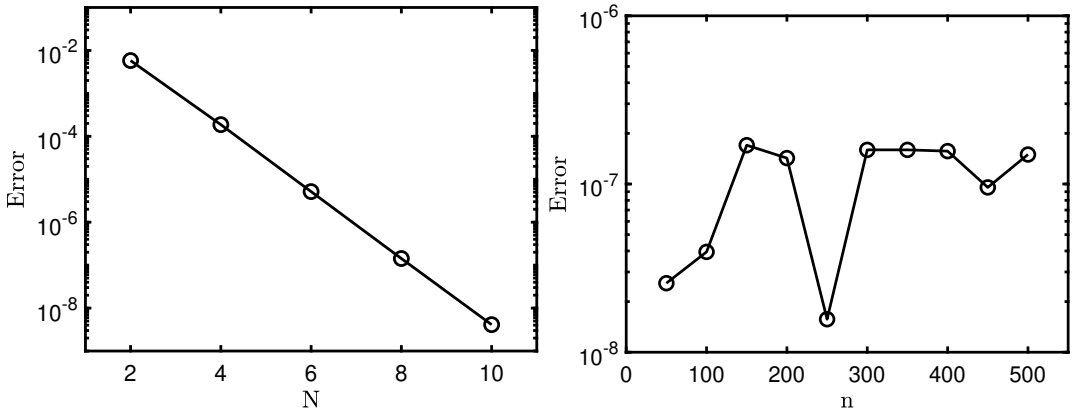
$$\frac{\mathrm{d}x}{\mathrm{d}t} = -0.5\cos^2(x).$$

We set $x(0) = \frac{\pi}{4}$ and $T = 20$ in this example. Despite the nonlinearity, the system has a closed-form solution $x(t) = \arctan(-0.5t + \tan(x_0))$. We aim to compute the solution at $T = 20$. The three experiments use the following parameters:

(a) test of N: $n = 200, r = \frac{\pi}{20}$;

(b) test of n: $N = 9, r = \frac{\pi}{20}$;
(c) test of r: $n = 200, N = 9$.

In all these tests, we set $\gamma = 0.2$. Figure 1 summarizes these results in plots (a), (b) and (c), respectively. The first test shows the exponential convergence of ASK with respect to N, which is similar to the conclusions in conventional spectral methods. Test (b) shows that the accuracy does not change monotonically as n varies given the parameter setting in this work. On the other hand, using no more than 100 check points is sufficient to obtain good accuracy. The last test illustrates that the accuracy shows a "V shape" with respect to the radius, i.e., r can not be too large or too small.



(a) Gauss-Lobatto points

(b) Check points

(c) Radius

Figure 1: Cosine Model: (a) testing number of Gauss-Lobatto points N; (b) testing number of checck points n; (c) testing radius r.

4.2. Lotka-Volterra model. The Lotka-Volterra equations model the interactive evolution of the population of prey and predators [2]. Specifically, it is defined by

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = 1.1x_1 - 0.4x_1x_2,$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = 0.1x_1x_2 - 0.4x_2.$$

We set $\boldsymbol{x}(0) = (10, 5)^\top$ and $T = 20$ in this example. The parameters used in three different tests are as follows:

(a) test of N: $n = 200, r = 1.5$;
(b) test of n: $N = 5, r = 1.5$;
(c) test of r: $n = 200, N = 5$.

In all the tests, γ is set to 0.5. Note that for multi-dimensional systems in the test of radius, all components share the same radius. Figure 2 presents the results of these tests. Similar to the cosine model, the error decreases exponentially with respect to N. The accuracy is quite stable with respect to the number of check points in this case. Furthermore, Figure 2c shows that the radius cannot not be too small as in the first example.

4.3. Simple pendulum. The simple pendulum is well studied in physics and mechanics. The movement of the pendulum is described by a second order ordinary differential equation,

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L}\sin(\theta).$$

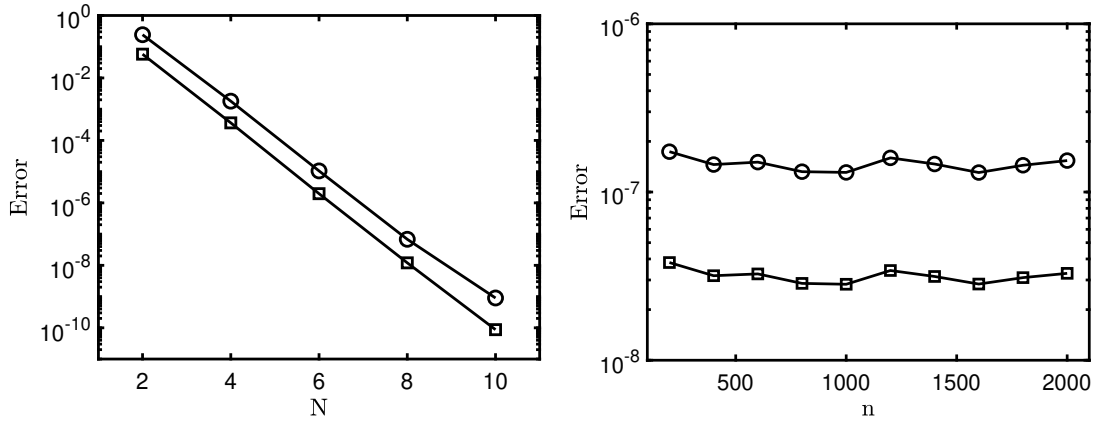
Here, θ is the displacement angle, and L denotes the length of the pendulum. The parameter g is the gravity acceleration. This second order equation can be converted to a two-dimensional first-order ODE system. To keep the notations consistent, we define $x_1 \coloneqq \theta$ and $x_2 \coloneqq \frac{d\theta}{dt}$. Also, we set $L = g = 9.8$ in our numerical experiments. Correspondingly,

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = x_2,$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = -\sin x_1.$$

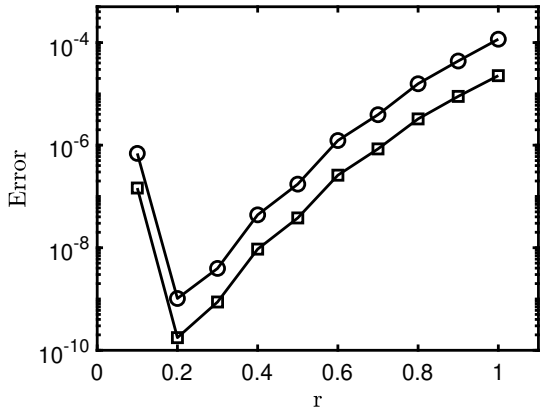
We set $\boldsymbol{x}(0) = (-\frac{\pi}{4}, \frac{\pi}{6})^\top$ and $T = 20$. The parameters in the three tests are as follows:

(a) test of N: $n = 200, r = \frac{\pi}{8}$;
(b) test of n: $N = 7, r = \frac{\pi}{8}$;
(c) test of r: $n = 200, N = 7$.

We set $\gamma = 0.2$ in all these tests. The resutls are presented in Figure 3. Again, we observe exponential convergence with respect to N in Figure 3a. Figure 3b implies that more check points can improve the accuracy but the difference is not very large. Figure 3c indicates there exists an "optimal" r as in the Lotka-Volterra example.



(a) Gauss-Lobatto points (b) Check points



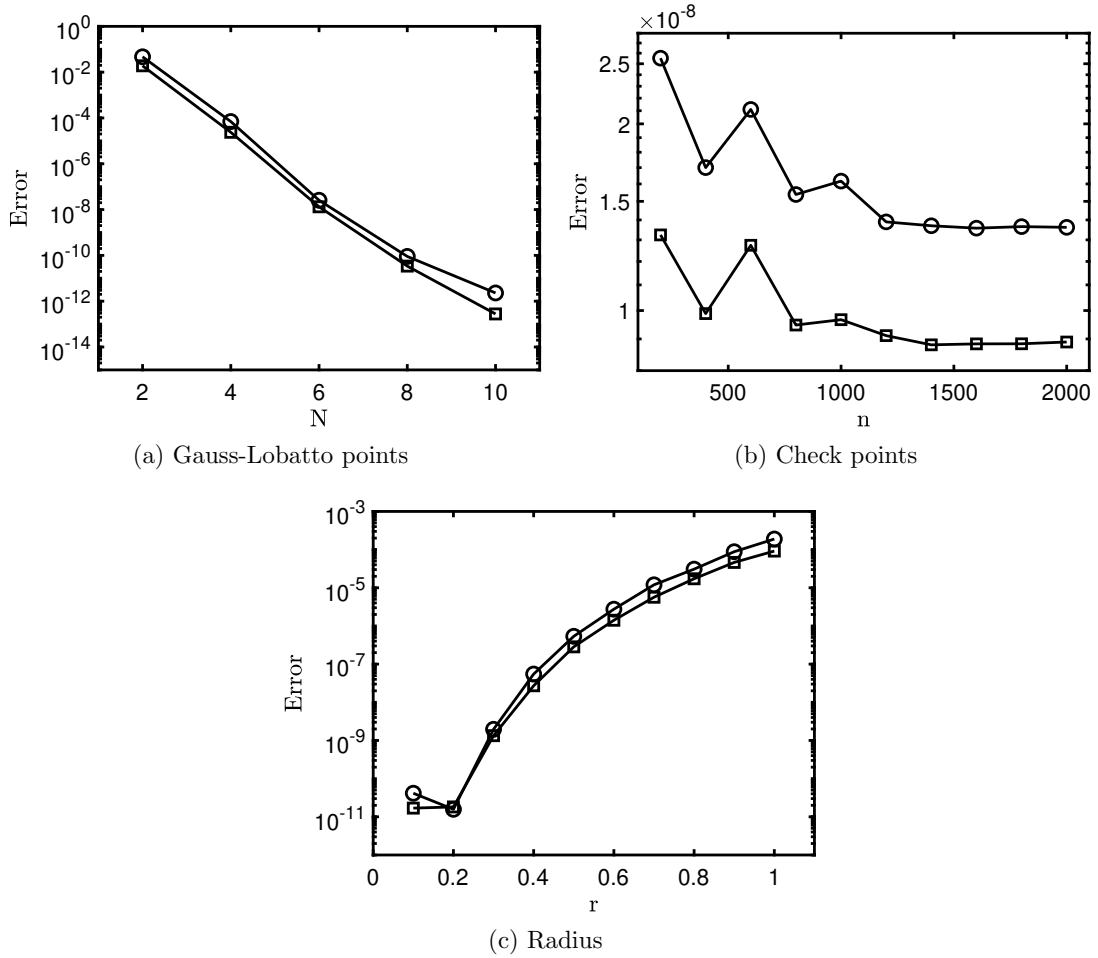
(c) Radius

Figure 2: Lotka-Volterra Model: (a) testing number of Gauss-Lobatto points N (total number of collocation points is $(N+1)^2$); (b) testing number of checck points n; (c) testing radius r. $\circ, \square$ denote x_1, x_2, respectively.

4.4. Limit cycle. The limit cycle is applied to model oscillatory systems in multiple research fields [32]. Here, we follow the definition,

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = -x_1 - x_2 + \frac{x_1}{\sqrt{x_1^2 + x_2^2}},$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = x_1 - x_2 + \frac{x_2}{\sqrt{x_1^2 + x_2^2}}.$$

The closed-form solution is $x_1(t) = \cos(t + \arcsin(x_1(0)))$ and $x_2(t) = \sin(t + \arcsin(x_2(0)))$. We set $\boldsymbol{x}(0) = (\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})^\top$ and $T = 20$ in this example. The parameters in the experiments



(a) Gauss-Lobatto points

(b) Check points

(c) Radius

Figure 3: Simple Pendulum: (a) testing number of Gauss-Lobatto points N (i.e., $(N+1)^2$ collocation points in total); (b) testing number of checck points n; (c) testing radius r. $\circ, \square$ denote x_1, x_2, respectively.

are specified as follows:

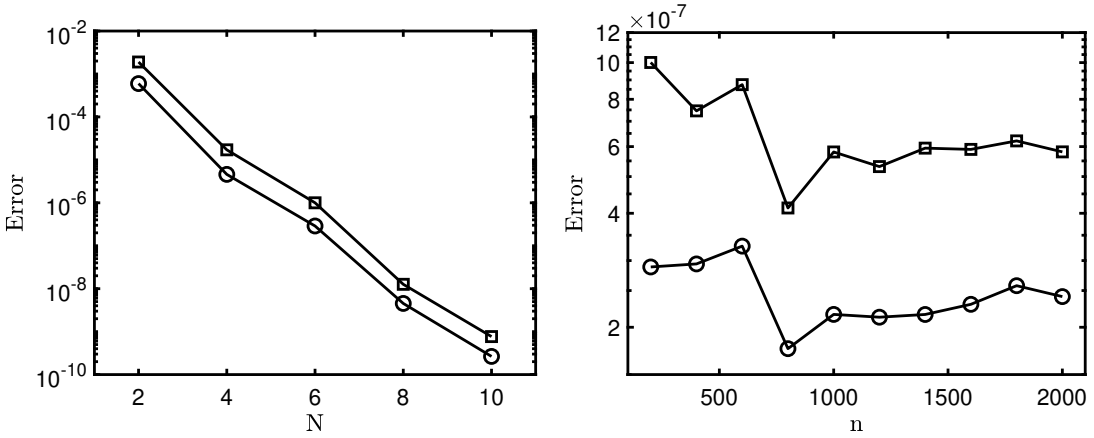
(a) test of N: $n = 200, r = \frac{\sqrt{2}}{6}$;

(b) test of n: $N = 7, r = \frac{\sqrt{2}}{6}$;

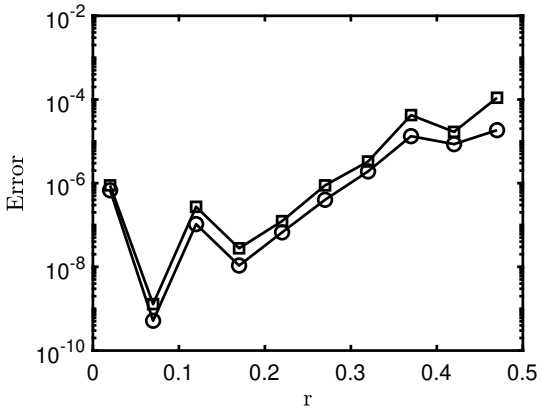
(c) test of r: $n = 200, N = 7$.

We set $\gamma = 0.2$ in all these tests. The results shown in Figure 4 reveal similar patterns to the results of the simple pendulum, except that a very small r can still lead to accurate results.

For this example, we also compared ASK with RK4 at various time within $[0, T]$. Given the closed-form solution $x_C(t)$, we computed the errors by $|x_{ASK}(t) - x_C(t)|$ and $|x_{RK4}(t) - x_C(t)|$.



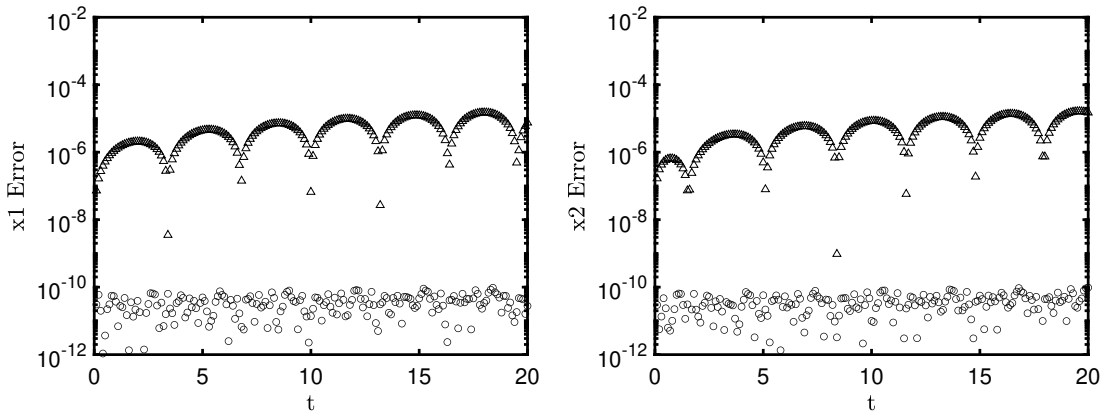
(a) Gauss-Lobatto points (b) Check points



(c) Radius

Figure 4: Limit Cycle: (a) testing number of Gauss-Lobatto points N (i.e., $(N+1)^2$ collocation points in total); (b) testing number of checck points n; (c) testing radius r. $\circ, \square$ denote x_1, x_2, respectively.

Here, RK4 employed $M = 200$ equidistant time points on $[0, T]$. As for ASK, we used $N = 9, r = \frac{\sqrt{2}}{8}, \gamma = 0.2$ and the check points are set to be the same as the time points in RK4. With this set of parameters, ASK constantly outperforms RK4 significantly, as illustrated in Figure 5. For both components x_1 and x_2, the errors of ASK remain almost constant at the level of 10^{-10}. In comparison, the error of RK4 exhibited a periodic pattern, rising slowly from 10^{-6} to 10^{-5}. Moreover, Figure 6 illustrates the evolution of the limit cycle model along time. The path decided by the two components elevates spirally as time goes. If seen from above, the cross section is an exact circle.



(a) Error of x_1 (b) Error of x_2

Figure 5: Errors of the limit cycle solution on $[0, T]$: ○ denotes ASK and △ denotes RK4.

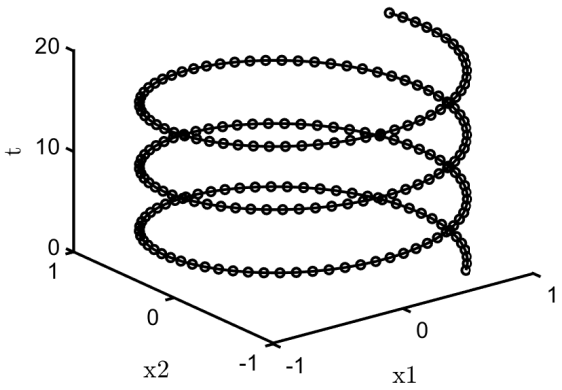


Figure 6: Limit cycle solution trajectory: ○ denotes ASK and – dentoes the closed-form solutions.

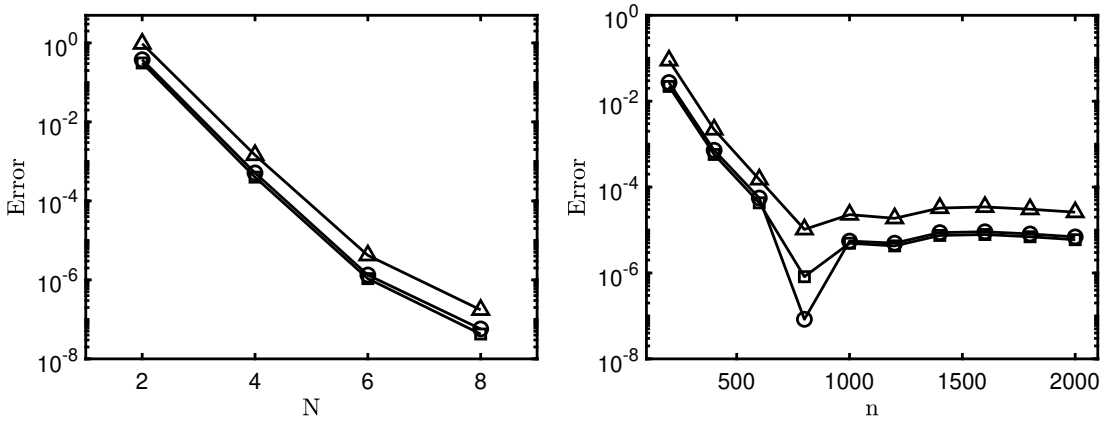
4.5. Kraichnan-Orszag model. The Kraichnan-Orszag model comes from the problem raised in [21]. This system is nonlinear and three-dimensional, defined by

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = x_2 x_3,$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = x_1 x_3,$$
$$\frac{\mathrm{d}x_3}{\mathrm{d}t} = -2x_1 x_2.$$

We set $\boldsymbol{x}(0) = (1, 2, -3)^\top$ and $T = 20$. In the three experiments, we employed the following parameters:

(a) test of N: $n = 400, r = 0.1$;
(b) test of n: $N = 3, r = 0.1$;
(c) test of r: $n = 400, N = 3$.

Also, in all the tests, we set $\gamma = 0.15$. The results are presented in Figure 7a. In particular, different from previous examples, Figure 7b demonstrates that n significantly influences the accuracy. This is because the Kraichnan-Orszag model exhibited strong oscillations, so it requires more frequent update of eigenpairs to guarantee high accuracy. Figure 7c indicates that there is an "optimal" r as in the Lokta-Volterra example.



(a) Gauss-Lobatto points

(b) Check points

(c) Radius

Figure 7: Kraichnan-Orszag model: (a) testing number of Gauss-Lobatto points N (i.e., $(N+1)^3$ collocation points in total); (b) testing number of checck points n; (c) testing radius r. $\circ, \square, \triangle$ denote x_1, x_2, x_3, respectively

4.6. Lorenz attractor. The Lorenz attractor was first introduced by Lorenz [15]. It is a highly chaotic system that models the turbulence in dynamic flows. The governing equations are as follows,

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = 10(x_2 - x_1),$$
$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = x_1(28 - x_3) - x_2,$$
$$\frac{\mathrm{d}x_3}{\mathrm{d}t} = x_1x_2 - 3x_3.$$

We set $\boldsymbol{x}(0) = (5,5,5)^\top$ and $T = 20$ in this example. Parameters used in the experiments are listed here:

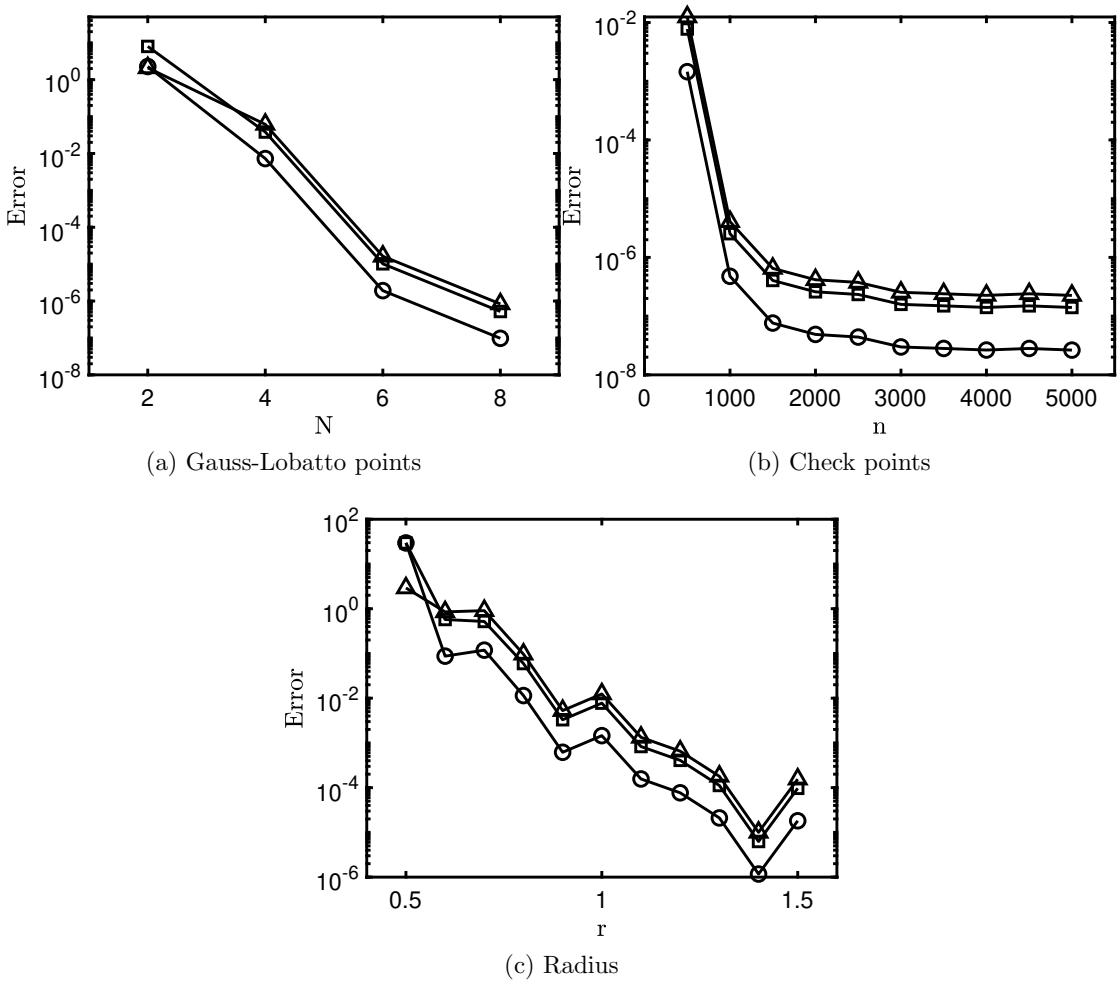
(a) test of N: $n = 500, r = 1$;
(b) test of n: $N = 5, r = 4$;
(c) test of r: $n = 500, N = 5$.

In all the tests, we set $\gamma = 0.5$. The results are summarized in Figure 8. As the Loren attractor exhibits chaotic behaviour, it requires a greater number of check points. Meanwhile, a relatively large radius favored the convergence of the algorithm.This is probably because the eigenfunctions need to be approximated in a larger neighborhood of the solution to include sufficient information of the dynamics.

Next we compare ASK and RK4 in a larger time span $[0, 20]$, where RK4 uses $M = 2000$ time steps, i.e., step size $\Delta t = 0.01$. Since the Lorenz attractor does not have closed-form solutions, RK9 is used to compute the reference. To guarantee accuracy, RK9 used step size $\Delta t = 0.001$, i.e., $M = 20000$ time steps. On the other hand, ASK was implemented with $n = 2000, N = 5, r = (1,1,1), \gamma = 0.75$. For the aforementioned reason, we required a large number of check points and small tolerance for the acceptable range. Figure 9 reveals the accuracy of ASK in all three components. However, unlike the limit cycle case, the error increases as time evolves. Although it rises to around 10^{-3} at $T = 20$, ASK still yields an acceptable accuracy for such a chaotic system. In comparison, RK4's error ascends to a level that makes it impractical. To obtain an insight of how the Lorenz system evolves, we plot each of its component in Figure 10. Up to time $T = 10$, solutions given by ASK, RK4, and RK9 almost coincide. Nevertheless, RK4 deviates from the other two completely starting at $t = 11$. The evolution vibrates violently and does not possess periodicity, which imposes difficulty on numerical solvers.

The chaos can also be observed in a three-dimensional graph depicting the trajectory, using the numerical solutions given by ASK. As in Figure 11, the lemniscate shape demonstrates the complexity of the system.

4.7. Computational complexity. By construction, the computational complexity of conventional ODE solvers that approximate time integral is $\mathcal{O}(M)$ where M is the number of time steps. In other words, it is M multiplied by a constant that represents the number of operations in each step and it varies according to the accuracy of the scheme. The computational complexity of ASK depends on the number of times that eigenfunctions are constructed (and corresponding eigenvalues as well as coefficients are computed). In this construction procedure, ASK needs to perform the eigendecomposition and solve a linear system. For $d = 1$



(a) Gauss-Lobatto points

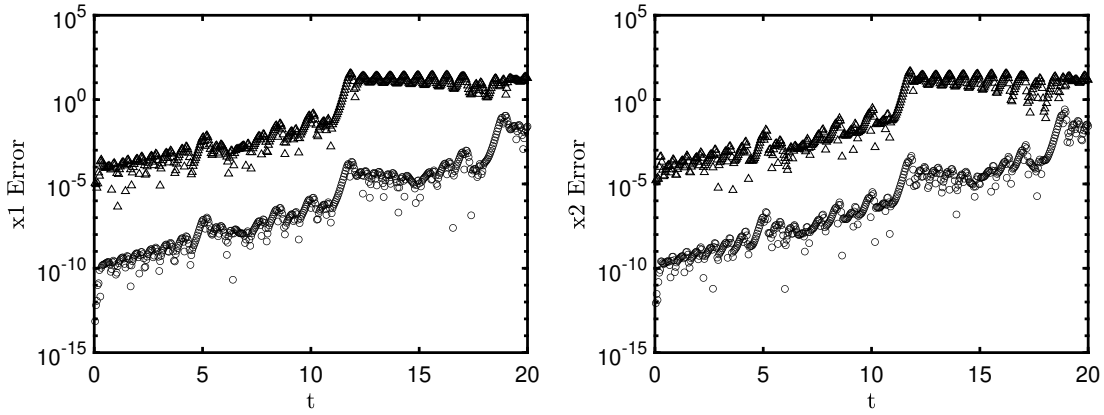
(b) Check points

(c) Radius

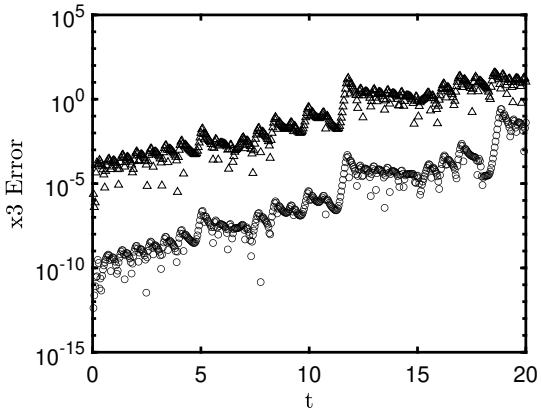
Figure 8: Loren attractor: (a) testing number of Gauss-Lobatto points N (i.e., $(N+1)^3$ collocation points in total); (b) testing number of checck points n; (c) testing radius r. $\circ, \square, \triangle$ denote x_1, x_2, x_3, respectively.

this is not costly because empirically we set $4 \leq N \leq 10$, and the size of matrix in the eigen-decomposition as well as the linear system is $N \times N$. But when $d > 1$, the complexity will increase exponentially with the dimension of the current setting because we use the tensor product rule to construct the collocation points and the matrix size is $(N+1)^d \times (N+1)^d$. Hence, ASK can be less efficient for high-dimensional systems.

However, compared with the Runge-Kutta methods, ASK has advantages when it is costly to evaluate the dynamics $\mathbf{f}$ and when ASK only needs a small number of updates. Here, we consider an evaluation of $\mathbf{f}$ as a function call, and compare the accuracy of ASK, RK4 and RK9



(a) Error of x_1 (b) Error of x_2

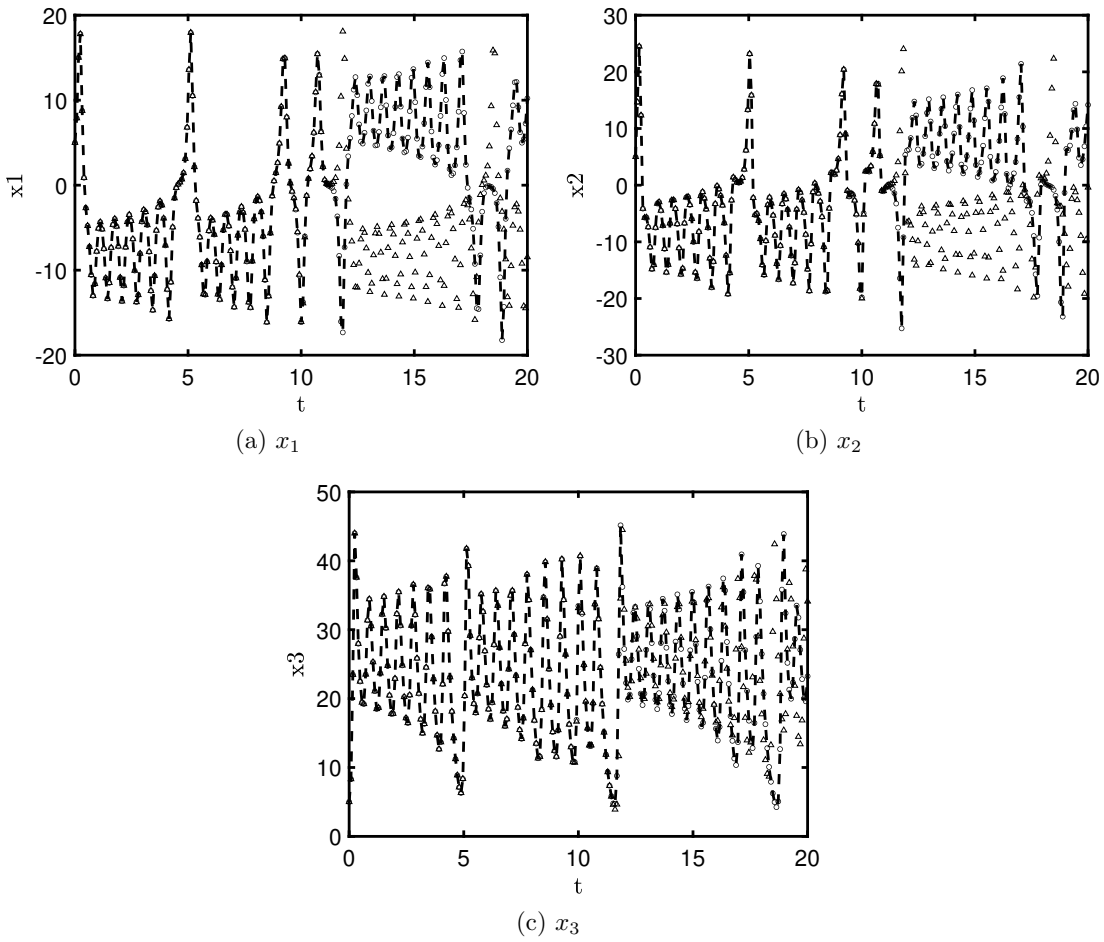


(c) Error of x_3

Figure 9: Error of Lorenz attractor solutions on $[0, T]$: ○ denotes ASK and △ dentoes RK4.

against number of function calls. The cosine model ($d = 1$) and the simple pendulum ($d = 2$) exemplify the comparison and give the results in Figure 12. Here, the error in the simple pendulum case was computed by $\sqrt{\frac{e_1^2+e_2^2}{2}}$, where e_1, e_2 are the errors in x_1, x_2, respectively. The parameters in ASK are $N = [4, 6, 8, 10], n = 50, r = \frac{\pi}{20}$ for the cosine model and $N = [2, 4, 6, 8], n = 60, r = \frac{\sqrt{2}}{12}$ for the simple pendulum. ASK is superior to the other two methods in the cosine model. For the simple pendulum, RK9 is the most efficient method, and ASK outperforms RK4 when number of function calls is beyond 2000.

5. Conclusion and Discussion. The ASK method uses the spectral-collocation (i.e., pseudo spectral) method in the state space instead of in time to solve nonlinear autonomous dynamical systems. It discretizes the generator of Koopman operator and employs the eigendecomposition to obtain approximations of the eigenfunctions and eigenvalues to construct solutions.



(a) x_1 (b) x_2

(c) x_3

Figure 10: Lorenz Attractor Evolution: ○ and △ denote ASK and RK4, respectively; −− denotes the reference solutions given by RK9

Therefore, like the spectral method, ASK is an expansion-based method to solve ODE systems, in which the basis functions in the expansion are approximated eigenfunctions of the Koopman operator. In each numerical example presented in this work, ASK exhibits *exponential convergence* as the conventional spectral method. Therefore, it is suitable for the circumstances where high-accuracy solutions are desired and $\mathbf{f}$ is expensive to evaluate. Different from existing ODE solvers that obtain solutions on mesh grids, ASK does not need a time mesh and can evaluate the solution at any time. Hence, the resolution of the time mesh which impacts the solutions of conventional ODE solvers like Runge-Kutta methods does not influence ASK.

In the ASK algorithm, adaptively updating the eigenfunction approximations in the neigh-

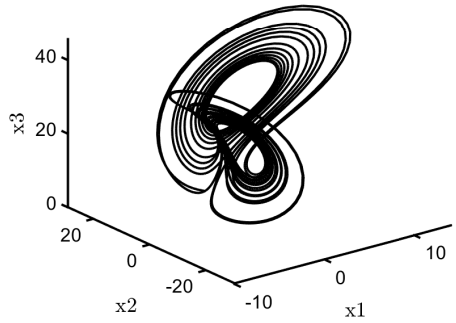
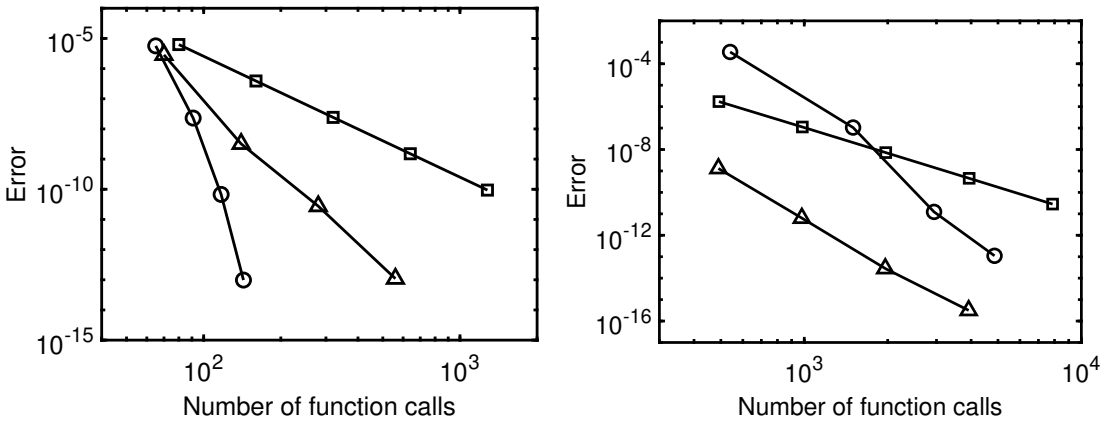


Figure 11: Lorenz Attractor 3D Visualization



(a) Cosine model (b) Simple Pendulum

Figure 12: Test of Complexity: ○, □, and △ denote ASK, RK4, and RK9, respectively.

borhood of the solution is necessary because it is challenging to obtain very accurate approximation of the eigenfunctions, eigenvalues and Koopman modes using the initial condition only, especially for highly nonlinear systems. When no information (e.g., range of states, regularity of the eigenfunctions) of the system is available *a priori*, the adaptivity criterion serves as a updating step based on "*posterior* error estimates". Furthermore, tunable parameters r and γ affect the accuracy as they are related to eigenfunction approximations and the adaptivity criterion. Numerical analysis based on the spectrum theorem as well as the spectral method is required to systematically understand the convergence and the impact of all parameters on the performance of ASK, which will be included in our future work.

Regarding the computational complexity, as indicated in subsection 4.7, ASK's efficiency decreases (compared with Runge-Kutta) as the system dimension increases since the ten-

sor product rule is applied to construct high-dimensional collocation points. A possible way of improving the efficiency is to leverage the sparse grid methods to construct collocation points, which has shown success in solving partial differential equations with the spectral method [27, 28]. Also, applying an anisotropic setting, e.g., different number of Gauss-Lobatto points, different radius, different γ in each direction, can potentially enhance the computational efficiency.

Appendix A. An example of the obervable.

As an example, we consider the following nonlinear dynamical system [3, 16]:

$$\begin{aligned}\frac{\mathrm{d}x_1}{\mathrm{d}t} &= \alpha x_1,\\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= \beta(x_2 - x_1^2).\end{aligned}$$

Here, α and β are the inherent parameters of the system. For such a system, appropriate observables lead to a closed-form solution. In particular, let $\boldsymbol{y} := (x_1, x_2, x_1^2)^\top$ be a three-dimensional observable. Then, the system can be converted to the following linear system,

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = \begin{bmatrix}\alpha & 0 & 0\\ 0 & \beta & -\beta\\ 0 & 0 & 2\alpha\end{bmatrix}\boldsymbol{y}.$$

For simplicity, assume $x_1(0) = x_2(0) = 1$. Then, we have the closed-form solution

$$\boldsymbol{y} = \begin{bmatrix}1\\0\\0\end{bmatrix}e^{\alpha t} + \frac{-2\alpha}{\beta - 2\alpha}\begin{bmatrix}0\\1\\0\end{bmatrix}e^{\beta t} + \begin{bmatrix}0\\ \frac{\beta}{\beta-2\alpha}\\ 1\end{bmatrix}e^{2\alpha t} = \begin{bmatrix}e^{\alpha t}\\ \frac{-2\alpha}{\beta-2\alpha}e^{\beta t} + \frac{\beta}{\beta-2\alpha}e^{2\alpha t}\\ e^{2\alpha t}\end{bmatrix}.$$

Equivalently,

$$x_1 = e^{\alpha t}, \qquad x_2 = \frac{-2\alpha}{\beta - 2\alpha}e^{\beta t} + \frac{\beta}{\beta - 2\alpha}e^{2\alpha t}.$$

Appendix B. An example pseudocode.

We demonstrate a pseudo code (in MATLAB) of solving $\frac{\mathrm{d}x}{\mathrm{d}t} = \cos^2(x)$, which summarizes the steps presented in subsection 3.1–subsection 3.3. The MATLAB code generating Chebyshev-Gauss-Lobatto points and the associated differentiation matrix can be found in [29].

```
f = @(x) cos(x).^2;  % Function f
x0 = pi/4;           % Initial condition
r = 0.1;             % Radius of the neighborhood (tunable)
N = 4;               % Number of collocation points (N+1 in total)
T = 5;               % Final time
% Generate collocation points and the differentiation matrix
% on [x0-r, x0+r]
[quad_pnt, diff_mat] = cheb(N, x0-r, x0+r);
% Compute eigenpairs of the Koopman operator
```

```
    K = diag(f(quad_pnt))*diff_mat;
    [eig_vec, eig_val] = eig(K, 'vector');
    % Compute coefficients (Koopman modes)
    coef = eig_vec\quad_pnt;
    % Construct solutions at time T
    sol = real(eig_vec(N/2,:).*coef'*exp(eig_val*T));
```

When the adaptive update in ASK is activated (see subsection 3.4), we only need to repeat this pseudocode (as a subroutine) with an updated initial condition x_0 and final time T.

Acknowledgments. We thank Professor Yue Yu, Hong Qian, and Yeonjoing Shin for fruitful discussions on the spectral method and properties of the Koopman operator.

REFERENCES

[1] T. Askham and J. N. Kutz, *Variable projection methods for an optimized dynamic mode decomposition*, SIAM Journal on Applied Dynamical Systems, 17 (2018), pp. 380–416.
[2] I. M. Bomze, *Lotka-Volterra equation and replicator dynamics: a two-dimensional classification*, Biological cybernetics, 48 (1983), pp. 201–211.
[3] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, *Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control*, PloS one, 11 (2016), p. e0150171.
[4] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, *Modern Koopman theory for dynamical systems*, arXiv preprint arXiv:2102.12086, (2021).
[5] M. Budišić and I. Mezić, *An approximate parametrization of the ergodic partition using time averaged observables*, in Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, IEEE, 2009, pp. 3162–3168.
[6] M. Budišić, R. Mohr, and I. Mezić, *Applied Koopmanism*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 22 (2012), p. 047510.
[7] J. C. Butcher and G. Wanner, *Runge-Kutta methods: some historical notes*, Applied Numerical Mathematics, 22 (1996), pp. 113–151.
[8] B. Fornberg, *A practical guide to pseudospectral methods*, no. 1, Cambridge university press, 1998.
[9] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb, *Spectral methods for time-dependent problems*, vol. 21, Cambridge University Press, 2007.
[10] G. Karniadakis and S. Sherwin, *Spectral/hp element methods for computational fluid dynamics*, OUP Oxford, 2005.
[11] B. O. Koopman, *Hamiltonian systems and transformation in Hilbert space*, Proceedings of the National Academy of Sciences of the United States of America, 17 (1931), p. 315.
[12] M. Korda, M. Putinar, and I. Mezić, *Data-driven spectral analysis of the Koopman operator*, Applied and Computational Harmonic Analysis, 48 (2020), pp. 599–629.
[13] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*, SIAM, 2016.
[14] J. N. Kutz, X. Fu, and S. L. Brunton, *Multiresolution dynamic mode decomposition*, SIAM Journal on Applied Dynamical Systems, 15 (2016), pp. 713–735.
[15] E. N. Lorenz, *Deterministic nonperiodic flow*, Journal of atmospheric sciences, 20 (1963), pp. 130–141.
[16] B. Lusch, J. N. Kutz, and S. L. Brunton, *Deep learning for universal linear embeddings of nonlinear dynamics*, Nature communications, 9 (2018), pp. 1–10.
[17] I. Mezić, *Spectral properties of dynamical systems, model reduction and decompositions*, Nonlinear Dynamics, 41 (2005), pp. 309–325.
[18] I. Mezić, *Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry*, Journal of Nonlinear Science, 30 (2020), pp. 2091–2145.
[19] H. Nakao and I. Mezić, *Spectral analysis of the Koopman operator for partial differential equations*,

Chaos: An Interdisciplinary Journal of Nonlinear Science, 30 (2020), p. 113131.
[20] J. Nathan Kutz, J. L. Proctor, and S. L. Brunton, *Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems*, Complexity, 2018 (2018).
[21] S. A. Orszag and L. Bissonnette, *Dynamical properties of truncated Wiener-Hermite expansions*, The Physics of Fluids, 10 (1967), pp. 2603–2613.
[22] J. Page and R. R. Kerswell, *Koopman analysis of Burgers equation*, Physical Review Fluids, 3 (2018), p. 071901.
[23] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Dynamic mode decomposition with control*, SIAM Journal on Applied Dynamical Systems, 15 (2016), pp. 142–161.
[24] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, *Spectral analysis of nonlinear flows*, Journal of fluid mechanics, 641 (2009), pp. 115–127.
[25] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, Journal of fluid mechanics, 656 (2010), pp. 5–28.
[26] J. Shen and T. Tang, *Spectral and high-order methods with applications*, Science Press Beijing, 2006.
[27] J. Shen and H. Yu, *Efficient spectral sparse grid methods and applications to high-dimensional elliptic problems*, SIAM Journal on Scientific Computing, 32 (2010), pp. 3228–3250.
[28] J. Shen and H. Yu, *Efficient spectral sparse grid methods and applications to high-dimensional elliptic equations ii. unbounded domains*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1141–A1164.
[29] L. N. Trefethen, *Spectral methods in MATLAB*, SIAM, 2000.
[30] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, *On dynamic mode decomposition: Theory and applications*, Journal of Computational Dynamics, 1 (2014), pp. 391–421.
[31] J. H. Verner, *Explicit Runge–Kutta methods with estimates of the local truncation error*, SIAM Journal on Numerical Analysis, 15 (1978), pp. 772–790.
[32] M. Vidyasagar, *Nonlinear systems analysis*, SIAM, 2002.
[33] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, *Extending data-driven Koopman analysis to actuated systems*, IFAC-PapersOnLine, 49 (2016), pp. 704–709.
[34] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition*, Journal of Nonlinear Science, 25 (2015), pp. 1307–1346.
[35] D. Wilson and J. Moehlis, *Isostable reduction with applications to time-dependent partial differential equations*, Physical Review E, 94 (2016), p. 012211.