SIAM J. SCI. COMPUT.
Vol. 46, No. 5, pp. A2925–A2950

THE SPARSE-GRID-BASED ADAPTIVE SPECTRAL KOOPMAN METHOD*

BIAN LI†, YUE YU‡, AND XIU YANG†

Abstract. The adaptive spectral Koopman (ASK) method was introduced to numerically solve autonomous dynamical systems that laid the foundation for numerous applications across different fields in science and engineering. Although ASK achieves high accuracy, it is computationally more expensive for multidimensional systems compared with conventional time integration schemes like Runge–Kutta. In this work, we combine the sparse grid and ASK to accelerate the computation for multidimensional systems. This sparse-grid-based ASK (SASK) method uses the Smolyak structure to construct multidimensional collocation points as well as associated polynomials that are used to approximate eigenfunctions of the Koopman operator of the system. In this way, the number of collocation points is reduced compared with using the tensor product rule. We demonstrate that SASK can be used to solve ordinary differential equations (ODEs) and partial differential equations (PDEs) based on their semidiscrete forms. Numerical experiments are illustrated to compare the performance of SASK and state-of-the-art ODE solvers.

Key words. dynamical systems, sparse grids, Koopman operator, partial differential equations, spectral-collocation method

MSC codes. 68Q25, 68R10, 68U05

DOI. 10.1137/23M1578292

1. Introduction. The Koopman operator [19] is an infinite-dimensional *linear* operator that describes the evolution of a set of observables. It provides a principled and often global framework to describe the dynamics of a finite-dimensional *nonlinear* system. Consequently, the Koopman operator approach to *nonlinear* dynamical systems has attracted considerable attention in recent years. One can define its eigenvalues, eigenfunctions, and modes and then use them to represent dynamically interpretable low-dimensional embeddings of high-dimensional state spaces to construct solutions through linear superposition [6]. In particular, 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 [29, 20, 31].

The Koopman operator provides powerful analytic tools to understand behaviors of dynamical systems 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 [28]. This approach has been applied to study ordinary differential equations

*Submitted to the journal's Numerical Algorithms for Scientific Computing section June 9, 2023; accepted for publication (in revised form) May 21, 2024; published electronically September 11, 2024.
https://doi.org/10.1137/23M1578292

Funding: The work of the first author was partially supported by (LANL LDRD). The work of the second author was supported by the NSF under award DMS-1753031 and the AFOSR grant FA9950-22-1-0197. The work of the third author 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) and by National Science Foundation (NSF) CAREER DMS-2143915.

†Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015 USA (bil215@lehigh.edu, xiy518@lehigh.edu).

‡Department of Mathematics, Lehigh University, Bethlehem, PA 18015 USA (yuy214@lehigh.edu).

(ODEs), partial differential equations (PDEs) [47, 23, 35, 31], dissipative dynamical systems [29], etc. Furthermore, novel numerical schemes, especially data-driven algorithms, motivated by or related to the Koopman operator have attracted much attention in the past decade. For example, the *dynamic mode decomposition* (DMD) [37, 38, 43, 36, 22, 23, 2, 11] and its variants like extended DMD (EDMD) [46, 25] use snapshots of a dynamical system to extract temporal features as well as correlated spatial activity. Subsequently, they can predict the behavior of the system in a short time. These approaches have been applied to design filters (e.g., [41, 33]), train neural networks (e.g., [10]), etc.

In [24] we proposed a novel numerical method based on the spectral-collocation method (i.e., the pseudospectral method) [12, 42] to implement the Koopman operator approach to solving nonlinear ODEs. This method leverages the differentiation matrix in spectral methods to approximate the generator of the Koopman operator and then conducts an eigendecomposition numerically to obtain eigenvalues and eigenvectors that approximate the 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 Koopman modes are approximated by computing eigenvalues, eigenvectors, and the observable based on the initial state. This approach is more efficient than the conventional ODE solvers such as Runge–Kutta and Adam–Bashforth for low-dimensional ODEs in terms of computational time, especially when evaluating the dynamics is costly [24]. This is because it allows evaluating the dynamics of the system at multiple collocation points simultaneously instead of computing them sequentially at different time steps as the aforementioned state-of-the-art ODE solvers do. In other words, ASK introduces a *new parallelization mechanism* for solving ODEs, which makes it more efficient than conventional approaches for some problems.

However, ASK's efficiency decreases as the system's dimension increases for it employs the tensor product rule to construct multidimensional collocation points and basis functions for polynomial interpolation. Consequently, the number of such points as well as basis functions increases exponentially. This number is associated with the size of the eigendecomposition problem and the linear system in the ASK scheme. Therefore, ASK is less efficient in multidimensional cases. To overcome this difficulty, we propose to combine the sparse grids method with ASK, wherein the Smolyak structure is applied to construct collocation points. This sparse-grid-based ASK (SASK) method reduces the number of collocation points and that of basis functions used in the vanilla ASK. Hence, the computational efficiency is enhanced. In numerical experiments, we demonstrate that SASK can solve ODEs and PDEs (based on their semidiscrete forms) accurately.

The paper is organized as follows. Section 2 introduces the background topics. A detailed discussion of the sparse-grid-based adaptive spectral Koopman method follows in section 3. Section 4 presents a numerical analysis to evaluate the convergence of ASK and SASK. We then show our numerical results in section 5. Finally, section 6 concludes the paper with a summary and further discussion.

2. Background.

2.1. Koopman operator.

Borrowing notations from [21], we consider an autonomous system described by the ODEs

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

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 aim to investigate the behavior of observables on the state space. For this purpose, 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 [28]). 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 [29].

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 semigroup.

2.2. Infinitesimal generator. The Koopman spectral theory [28, 37] unveils 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 infinitesimal generator (or generator for brevity) of the Koopman operator. Specifically, for any smooth observable function g, the generator of the Koopman operator $\mathcal{K}_t$, denoted as $\mathcal{K}$, is given by

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

which leads to

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

Denoting by φ an eigenfunction of $\mathcal{K}$ and by λ the eigenvalue associated with φ, we have $\mathcal{K}\varphi(\boldsymbol{x}) = \lambda\varphi(\boldsymbol{x})$, and hence $\lambda\varphi(\boldsymbol{x}) = \mathcal{K}\varphi(\boldsymbol{x}) = \frac{\mathrm{d}\varphi(\boldsymbol{x})}{\mathrm{d}t}$. This indicates that $\varphi(\boldsymbol{x}(t_0 + t)) = e^{\lambda t}\varphi(\boldsymbol{x}(t_0))$, i.e.,

$$\mathcal{K}_t\varphi(\boldsymbol{x}(t_0)) = e^{\lambda t}\varphi(\boldsymbol{x}(t_0)). \tag{2.5}$$

Therefore, φ is an eigenfunction of $\mathcal{K}_t$ associated with eigenvalue λ. Note that, following notations in literature, we consider the eigenpair for $\mathcal{K}_t$ 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., $g(\boldsymbol{x}) = \sum_j^\infty c_j\varphi_j(\boldsymbol{x})$; then

$$\mathcal{K}_t[g(\boldsymbol{x}(t_0))] = \mathcal{K}_t\left[\sum_j^\infty c_j\varphi_j(\boldsymbol{x}(t_0))\right] = \sum_j^\infty c_j\mathcal{K}_t[\varphi_j(\boldsymbol{x}(t_0))]. \tag{2.6}$$

Hence,

$$g(\boldsymbol{x}(t_0+t)) = \sum_j^{\infty} c_j \varphi_j(\boldsymbol{x}(t_0)) e^{\lambda_j t}. \tag{2.7}$$

Similarly, for 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

$$\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x})}{\mathrm{d}t} = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{K}g_1(\boldsymbol{x}) \\ \mathcal{K}g_2(\boldsymbol{x}) \\ \vdots \\ \mathcal{K}g_d(\boldsymbol{x}) \end{bmatrix} = \sum_j^{\infty} \lambda_j \varphi_j(\boldsymbol{x}) \boldsymbol{c}_j, \tag{2.8}$$

where $\boldsymbol{c}_j \in \mathbb{C}^d$ is called the jth Koopman mode with $\boldsymbol{c}_j := (c_j^1, c_j^2, \ldots, c_j^d)^\top$.

The ASK method uses the following truncated form of (2.7):

$$g(\boldsymbol{x}(t_0+t)) = \sum_j^{\infty} c_j \varphi_j(\boldsymbol{x}(t_0)) e^{\lambda_j t} \approx \sum_{j=0}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) e^{\tilde{\lambda}_j t} \tag{2.9}$$

for $d = 1$. Here, φ_j are approximated by Nth order interpolation polynomials φ_j^N, where N is a positive integer. Also, λ_j and c_j are approximated by $\tilde{\lambda}_j$ and $\tilde{c}_j$ [24]. For $d > 1$, φ_j^N is constructed by the tensor product rule with one-dimensional interpolation polynomials.

3. Sparse-grid-based adaptive spectral Koopman method. As mentioned in [24], ASK suffers from the curse of dimensionality as we approach high-dimensional systems. It is not surprising to see this phenomenon in numerical integration and interpolation on multidimensional domains when the tensor product rule is used to construct high-dimensional quadrature points. Specifically, if N denotes the number of points for one dimension and d denotes the number of dimensions, the tensor product rule gives a domain containing N^d points, which quickly grows prohibitive with d.

The sparse grid method is one of the most effective approaches to overcoming the aforementioned difficulties to a certain extent as it needs significantly fewer points in the computation. This method is also known as the Smolyak grid (or Smolyak's construction) in the name of Sergei A. Smolyak [40]. A series of seminal works further studied the properties of the sparse grid method and completed the framework [7, 16, 14, 48]. The full N^d-grid is a direct consequence of the tensor product of the points in each dimension, while the sparse grid method chooses only a subset of these grid points so that the total number increases much more slowly in d. As shown by Zenger [48], the total number of points is polynomial in d. This drastically reduces the computation complexity, enabling a more efficient variant of ASK. In this section, we introduce the sparse-grid-based adaptive spectral Koopman (SASK) method, an improved version of ASK for multidimensional problems.

3.1. Sparse grids for interpolation. The idea of sparse grids is that some grid points contribute more than the others in the numerical approximation. Thus, it does not undermine the interpolation if only a subset of the important grid points is utilized. In fact, the order of the error only increases slightly [48]. The polynomial interpolation in SASK borrows the ideas from [18], which leveraged Chebyshev extreme points to generate the sparse grids and Chebyshev polynomials to construct the basis functions.

Following a similar structure, this subsection first discusses the generation of the sparse grid. Then, the basis function interpolation is explained, followed by the computation of the coefficients in the linear combination of the basis functions.

3.1.1. Sparse grid construction. In this work, the points of a sparse grid are based on the extreme points of the Chebyshev polynomials. Specifically, denote $\xi_j = \cos\left(\frac{j\pi}{n_e-1}\right)$ for $j \in \{0,1,\dots,n_e-1\}$, where n_e is an integer. The construction of the sparse grids in a multidimensional domain builds on the unidimensional set of Chebyshev points that satisfy the Smolyak rule.

Let $\{\mathcal{N}_i\}_{i\geq 1}$ be a sequence of sets which contain the Chebyshev points such that the number of points in set i is $M(i) = 2^{i-1}+1$ for $i \geq 2$ and $M(1) = 1$, and that $\mathcal{N}_i \subset \mathcal{N}_{i+1}$. Then,

$$\mathcal{N}_1 = \{0\}, \quad \mathcal{N}_2 = \{0,-1,1\}, \quad \mathcal{N}_3 = \left\{0,-1,1,-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right\}.$$

To construct the sparse grid, we need another parameter, the approximation level $\kappa \in \mathbb{Z}_{\geq 0}$. This parameter controls the number of points n_p in one dimension, thus further dictating the overall degree of approximation. In particular, $n_p = 2^\kappa + 1$ when $\kappa \geq 1$ and $n_p = 1$ when $\kappa = 0$. Let i_j denote the index of the set in dimension j. Then, the Smolyak rule states that

$$d \leq \sum_{j=1}^{d} i_j \leq d + \kappa.$$

Here, we show an example with $d = 2, \kappa = 2$. In this case, $2 \leq i_1 + i_2 \leq 4$, and hence the possible combinations are as follows:

$$\begin{array}{lll} (1)\ i_1=1, i_2=1, & (2)\ i_1=2, i_2=1, & (3)\ i_1=3, i_2=1, \\ (4)\ i_1=1, i_2=2, & (5)\ i_1=1, i_2=3, & (6)\ i_1=2, i_2=2. \end{array} \tag{3.1}$$

Let $\mathcal{S}(\cdot,\cdot)$ be the tensor product of two sets of points. Then, the combinations (i_1,i_2) in (3.1) provide $\mathcal{S}(\mathcal{N}_{i_1},\mathcal{N}_{i_2})$. For example, $\mathcal{S}(\mathcal{N}_1,\mathcal{N}_3) = \{(0,0),(0,-1),(0,1),(0,-\frac{\sqrt{2}}{2}),(0,\frac{\sqrt{2}}{2})\}$. In this way, the sparse grid can be constructed as the union of $\mathcal{S}(\mathcal{N}_{i_1},\mathcal{N}_{i_2})$. The illustration of the sparse grids and its comparison with the full grids are enclosed in Appendix A.

By construction, there are repetitions of points in the union of $\mathcal{S}(\mathcal{N}_{i_1},\mathcal{N}_{i_2})$ since the sets are nested. For example, $\mathcal{S}(\mathcal{N}_1,\mathcal{N}_2) \cap \mathcal{S}(\mathcal{N}_1,\mathcal{N}_3) = \mathcal{S}(\mathcal{N}_1,\mathcal{N}_2)$. Hence, a more concise way to construct sparse grids is to apply disjoint sets [14, 18]. Denote $\mathcal{A}_i := \mathcal{N}_i \backslash \mathcal{N}_{i-1}$ for $i = 2,3,\dots$ and $\mathcal{A}_1 := \mathcal{N}_1$. Then, we have $\mathcal{A}_i \cap \mathcal{A}_{j\neq i} = \emptyset$. The number of points in $\mathcal{A}_i$ is computed by $\bar{M}(i) = M(i) - M(i-1) = 2^{i-2}$ for $i \geq 3$, $\bar{M}(1) = 1$, and $\bar{M}(2) = 2$. Specifically,

$$\mathcal{A}_1 = \{0\}, \quad \mathcal{A}_2 = \{-1,1\}, \quad \mathcal{A}_3 = \left\{-\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}\right\}.$$

Subsequently, we use the union of $\mathcal{S}(\mathcal{A}_{i_1},\mathcal{A}_{i_2})$ to construct sparse grids. By construction, $\bigcup_{i_1,i_2} \mathcal{S}(\mathcal{N}_{i_1},\mathcal{N}_{i_2}) = \bigcup_{i_1,i_2} \mathcal{S}(\mathcal{A}_{i_1},\mathcal{A}_{i_2})$.

3.1.2. Polynomial interpolation. We aim to approximate a smooth multivariate function $h(\boldsymbol{x})$ with a linear combination of polynomials that serve as the basis functions. Here, we choose the Chebyshev polynomials of the first kind to be the

univariate basis functions. It follows to construct multivariate basis functions with the tensor product of the univariate basis functions. The Chebyshev polynomials of the first kind T_k are given by a recurrence relation: $T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$ with $T_0(x) = 1, T_1(x) = x$, and $x \in [-1, 1]$. Hence, the basis functions used are $\psi_1(x) = 1, \psi_2(x) = x, \psi_3(x) = 2x^2 - 1, \psi_4(x) = 4x^3 - 3x$, and so on. Corresponding to the sets $\{\mathcal{A}_i\}$, we define the disjoint sets of unidimensional basis functions $\{\mathcal{F}_i\}$ by

$$\mathcal{F}_1 = \{\psi_1(x)\}, \quad \mathcal{F}_2 = \{\psi_2(x), \psi_3(x)\}, \quad \mathcal{F}_3 = \{\psi_4(x), \psi_5(x)\}.$$

Let $(x_1, x_2) \in [-1, 1] \times [-1, 1]$. By the Smolyak rule, the example above has the following basis function tensor products:

(1) $\mathcal{S}(\mathcal{F}_1, \mathcal{F}_1) = \{\psi_1(x_1)\psi_1(x_2)\}$,
(2) $\mathcal{S}(\mathcal{F}_2, \mathcal{F}_1) = \{\psi_2(x_1)\psi_1(x_2), \psi_3(x_1)\psi_1(x_2)\}$,
(3) $\mathcal{S}(\mathcal{F}_3, \mathcal{F}_1) = \{\psi_4(x_1)\psi_1(x_2), \psi_5(x_1)\psi_1(x_2)\}$,
(4) $\mathcal{S}(\mathcal{F}_1, \mathcal{F}_2) = \{\psi_1(x_1)\psi_2(x_2), \psi_1(x_1)\psi_3(x_2)\}$,
(5) $\mathcal{S}(\mathcal{F}_1, \mathcal{F}_3) = \{\psi_1(x_1)\psi_4(x_2), \psi_1(x_1)\psi_5(x_2)\}$,
(6) $\mathcal{S}(\mathcal{F}_2, \mathcal{F}_2) = \{\psi_2(x_1)\psi_2(x_2), \psi_2(x_1)\psi_3(x_2), \psi_3(x_1)\psi_2(x_2), \psi_3(x_1)\psi_3(x_2)\}$.

The union of these $\mathcal{S}(\mathcal{F}_{i_1}, \mathcal{F}_{i_2})$ forms the basis functions for the polynomial interpolation.

Suppose the total number of sparse grid points is N. Then, the total number of basis functions is also N, and we denote them as $\Psi_l(\boldsymbol{x})$, ordering them with index l. For example, $\Psi_1(\boldsymbol{x}) = \psi_1(x_1)\psi_1(x_2), \Psi_2(\boldsymbol{x}) = \psi_2(x_1)\psi_1(x_2)$, and so on. It then remains to approximate $h(\boldsymbol{x})$ as the following linear combination:

$$h(\boldsymbol{x}) \approx h^N(\boldsymbol{x}) = \sum_{l=1}^{N} w_l \Psi_l(\boldsymbol{x}),$$

where w_l denotes the unknown coefficients. Given the grid points $\{\boldsymbol{\xi}_l\}_{l=1}^N$, we can write

$$\begin{bmatrix} h^N(\boldsymbol{\xi}_1) \\ h^N(\boldsymbol{\xi}_2) \\ \vdots \\ h^N(\boldsymbol{\xi}_N) \end{bmatrix} = \begin{bmatrix} \Psi_1(\boldsymbol{\xi}_1) & \Psi_2(\boldsymbol{\xi}_1) & \dots & \Psi_N(\boldsymbol{\xi}_1) \\ \Psi_1(\boldsymbol{\xi}_2) & \Psi_2(\boldsymbol{\xi}_2) & \dots & \Psi_N(\boldsymbol{\xi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_1(\boldsymbol{\xi}_N) & \Psi_2(\boldsymbol{\xi}_N) & \dots & \Psi_N(\boldsymbol{\xi}_N) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} \tag{3.2}$$

as $\boldsymbol{h}^N = \mathbf{M}\boldsymbol{w}$, where $\boldsymbol{h}^N = \left(h^N(\boldsymbol{\xi}_1), \dots, h^N(\boldsymbol{\xi}_N)\right)^\top, \boldsymbol{w} = (w_1, \dots, w_N)^\top$, and $M_{ij} = \Psi_j(\boldsymbol{\xi}_i)$. Here, matrix $\mathbf{M}$ is full-ranked due to the orthogonality of Chebyshev polynomials. Vector $\boldsymbol{w}$ can be obtained by $\boldsymbol{w} = \mathbf{M}^{-1}\boldsymbol{h}^N$ when $\boldsymbol{h}^N$ and $\mathbf{M}$ are known.

3.2. Finite-dimensional approximation. To leverage the properties of the Koopman operator for solving dynamical systems, we intend to find the approximation of (2.7) as

$$g(\boldsymbol{x}(t + t_0)) \approx g_N(\boldsymbol{x}(t + t_0)) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) e^{\tilde{\lambda}_j t}, \tag{3.3}$$

where $\tilde{c}_j$ is the approximate Koopman mode, $\tilde{\lambda}_j$ is the approximate eigenvalue, and φ_j^N is the polynomial approximation of eigenfunction φ_j. Without loss of generality,

we will assume that $t_0 = 0$ and denote $\boldsymbol{x}_0 := \boldsymbol{x}(t_0) = \boldsymbol{x}(0)$. The property of the infinitesimal generator (2.4) leads to $\mathcal{K}\varphi(\boldsymbol{x}) = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \cdot \nabla\varphi(\boldsymbol{x})$ for any eigenfunction φ. Since $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathbf{f}(\boldsymbol{x})$, we have

$$\mathcal{K}\varphi = \mathbf{f} \cdot \nabla\varphi = f_1 \frac{\partial\varphi}{\partial x_1} + f_2 \frac{\partial\varphi}{\partial x_2} + \ldots + f_d \frac{\partial\varphi}{\partial x_d}. \tag{3.4}$$

The polynomial approximation φ_j^N in (3.3) can be obtained based on (3.4).

Consider the following polynomial approximation of an eigenfunction:

$$\varphi(\boldsymbol{x}) \approx \varphi^N(\boldsymbol{x}) = \sum_{l=1}^{N} w_l \Psi_l(\boldsymbol{x}).$$

It then follows that

$$\frac{\partial\varphi(\boldsymbol{x})}{\partial x_i} \approx \frac{\partial\varphi^N(\boldsymbol{x})}{\partial x_i} = \sum_{l=1}^{N} w_l \frac{\partial\Psi_l(\boldsymbol{x})}{\partial x_i} \quad \forall i.$$

Denote the sparse grid points in $\mathbb{R}^d$ by $\{\boldsymbol{\xi}_l\}_{l=1}^N$, where $\boldsymbol{\xi}_l := (\xi_{l_1}, \xi_{l_2}, \ldots, \xi_{l_d}) \in \mathbb{R}^d$. Replacing h in (3.2) with φ, we have $\boldsymbol{\varphi}^N = \mathbf{M}\boldsymbol{w}$, where $\boldsymbol{\varphi}^N$ is the vector of φ evaluated at $\boldsymbol{\xi}_l$. Accordingly, letting matrix $\mathbf{G}_i$ be $\partial_{x_i}\Psi_l(\boldsymbol{x})$ evaluated at the sparse grids points, we have

$$\mathbf{G}_i = \begin{bmatrix} \frac{\partial\Psi_1}{\partial x_i}(\boldsymbol{\xi}_1) & \frac{\partial\Psi_2}{\partial x_i}(\boldsymbol{\xi}_1) & \cdots & \frac{\partial\Psi_N}{\partial x_i}(\boldsymbol{\xi}_1) \\ \frac{\partial\Psi_1}{\partial x_i}(\boldsymbol{\xi}_2) & \frac{\partial\Psi_2}{\partial x_i}(\boldsymbol{\xi}_2) & \cdots & \frac{\partial\Psi_N}{\partial x_i}(\boldsymbol{\xi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\Psi_1}{\partial x_i}(\boldsymbol{\xi}_N) & \frac{\partial\Psi_2}{\partial x_i}(\boldsymbol{\xi}_N) & \cdots & \frac{\partial\Psi_N}{\partial x_i}(\boldsymbol{\xi}_N) \end{bmatrix}.$$

Then, $\partial_{x_i}\boldsymbol{\varphi}^N = \mathbf{G}_i\boldsymbol{w}$, where $(\partial_{x_i}\varphi^N)_l := \frac{\partial\Psi}{\partial x_i}(\boldsymbol{\xi}_l)$. Let $\mathbf{K} \in \mathbb{R}^{N\times N}$ be the finite-dimensional approximation of $\mathcal{K}$. Given the dynamics $\mathbf{f} = [f_1, f_2, \ldots, f_d]^\top$, (3.4) implies

$$\mathbf{K}\boldsymbol{\varphi}^N = \sum_{i}^{d} diag\big(f_i(\boldsymbol{\xi}_1), \ldots, f_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i\boldsymbol{w}. \tag{3.5}$$

3.3. Eigendecomposition. With the discretized Koopman operator, we intend to obtain the eigenfunction values using the eigendecomposition. One can formulate the eigenvalue problem $\mathcal{K}\varphi_j = \lambda_j\varphi_j$, where (φ_j, λ_j) is an eigenpair of $\mathcal{K}$. Correspondingly, the discrete eigenvalue problem is $\mathbf{K}\boldsymbol{\varphi}_j^N = \lambda_j\boldsymbol{\varphi}_j^N$. By (3.5) and $\boldsymbol{\varphi}_j^N = \mathbf{M}\boldsymbol{w}_j$, we have

$$\sum_{i}^{d} diag\big(f_i(\boldsymbol{\xi}_1), \ldots, f_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i\boldsymbol{w}_j = \tilde{\lambda}_j\mathbf{M}\boldsymbol{w}_j. \tag{3.6}$$

Let $\mathbf{U} := \sum_i^d diag\big(f_i(\boldsymbol{\xi}_1), \ldots, f_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i$; $\mathbf{U}\boldsymbol{w}_j = \tilde{\lambda}_j\mathbf{M}\boldsymbol{w}_j$ is a generalized eigenvalue problem, from which we solve for $\boldsymbol{w}_j$. For compactness, we write this in the matrix form

$$\mathbf{U}\mathbf{W} = \mathbf{M}\mathbf{W}\boldsymbol{\Lambda}, \tag{3.7}$$

where $\mathbf{W} := \begin{bmatrix}\boldsymbol{w}_1 & \boldsymbol{w}_2 & \ldots & \boldsymbol{w}_N\end{bmatrix}, \boldsymbol{\Lambda} := \mathrm{diag}(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_N)$. Then, the matrix of eigenfunctions can be defined by $\boldsymbol{\Phi}^N := \mathbf{M}\mathbf{W}$, whose jth column is $\boldsymbol{\varphi}_j^N$.

We note that SASK requires solving a generalized eigenvalue problem while ASK uses a standard eigendecomposition. This is because matrix $\mathbf{M}$ is an identity matrix $\mathbf{I}$ in ASK as it uses a Lagrange polynomial for the interpolation, which indicates that $\Psi_i(\boldsymbol{\xi}_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta function. Thus, in this setting, the generalized eigenvalue problem (3.7) is degenerated as $\mathbf{UW} = \mathbf{W\Lambda}$. Further, by construction, $\mathbf{G}_i = \mathbf{D}_i\mathbf{M}$, where $\mathbf{D}_i$ is the differentiation matrix in the ith direction. This formula reduces to $\mathbf{G}_i = \mathbf{D}_i$ when $\mathbf{M} = \mathbf{I}$ in ASK (see [24]). On the other hand, SASK uses a more general setting for the interpolation, i.e., Ψ_i are not necessarily Lagrange polynomials. Therefore, $\mathbf{M} \neq \mathbf{I}$ and the differentiation matrices $\mathbf{D}_i$ need to be obtained by solving a linear system. Instead of computing $\mathbf{D}_i$ explicitly, we compute $\mathbf{G}_i$ in SASK.

3.4. Constructing the solution. The eigendecomposition yields eigenfunction values φ_j^N at the sparse grid points $\boldsymbol{\xi}_l$. By construction, the central point of the domain is also the first sparse grid point generated. For example, $(0, 0, \ldots, 0) = \boldsymbol{\xi}_1$ for a multidimensional domain $[-1, 1]^d$. Hence, to avoid interpolating the eigenfunction when $\boldsymbol{x}_0 \notin \{\boldsymbol{\xi}_l\}$, we propose to construct a neighborhood of $\boldsymbol{x}_0$ defined by $[\boldsymbol{x}_0 - \boldsymbol{r}, \boldsymbol{x}_0 + \boldsymbol{r}]$, where $\boldsymbol{r} = (r_1, r_2, \ldots, r_d)^\top$ is the radius. Equivalently, the neighborhood in dimension i is $[x_0^i - r_i, x_0^i + r_i]$. For simplicity, we apply the isotropic setting with $r := r_1 = r_2 = \cdots = r_d$ in this work, but we emphasize that it is not necessary, and that the anisotropic setting might be more effective. Therefore, the observable of the dynamical system is constructed as

$$g_N(\boldsymbol{x}(t)) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}_0) e^{\tilde{\lambda}_j t}. \tag{3.8}$$

Setting $t = 0$, we compute the approximate Koopman modes $\tilde{c}_j$ using the following equation:

$$g(\boldsymbol{x}_0) \approx g_N(\boldsymbol{x}_0) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}_0),$$

which must be satisfied for different initial conditions in the neighborhood of $\boldsymbol{x}_0$. Thus, by considering all sparse grid points as different initial conditions, we have

$$g(\boldsymbol{\xi}_l) \approx g_N(\boldsymbol{\xi}_l) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{\xi}_l), \quad l = 1, 2, \ldots, N.$$

These formulas can be summarized in a matrix form by defining the matrix of the sparse grid as

$$\boldsymbol{\Xi} := \begin{bmatrix} (\boldsymbol{\xi}_1)^\top \\ (\boldsymbol{\xi}_2)^\top \\ \vdots \\ (\boldsymbol{\xi}_N)^\top \end{bmatrix} = \begin{bmatrix} \xi_{11} & \xi_{12} & \cdots & \xi_{1d} \\ \xi_{21} & \xi_{22} & \cdots & \xi_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{N1} & \xi_{N2} & \cdots & \xi_{Nd} \end{bmatrix},$$

and denoting column i of the matrix by $\boldsymbol{\Xi}_i$. If we choose the vector-valued observable $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x}$, then the Koopman modes must satisfy $\boldsymbol{\Phi}^N \boldsymbol{c}_i = \boldsymbol{\Xi}_i$ for all i. The Koopman modes are computed by solving these linear systems. In a more compact form, $\boldsymbol{\Phi}^N \mathbf{C} = \boldsymbol{\Xi}$. In particular, $\boldsymbol{c}_i$ is column i of the matrix $\mathbf{C} = (c_{ji})$, containing the Koopman modes for dimension i.

Finally, $\boldsymbol{\xi}_1 = \boldsymbol{x}_0$ by construction, and hence, $\varphi_j^N(\boldsymbol{x}_0)$ is the first element of vector $\boldsymbol{\varphi}_j^N$, denoted by $(\boldsymbol{\varphi}_j^N)_1$. Therefore, the solution of the dynamical system is constructed as

$$\boldsymbol{x}(t) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}_0) e^{\tilde{\lambda}_j t} = \sum_{j=1}^{N} \tilde{c}_j (\boldsymbol{\varphi}_j^N)_1 e^{\tilde{\lambda}_j t}, \tag{3.9}$$

wherein the observable function is identity.

3.5. Adaptivity. Due to the finite-dimensional approximation of $\mathcal{K}$ and local approximation (in the neighborhood of $\boldsymbol{x}_0$) of φ, the accuracy of the solution decays as the system evolves in time. This is particularly the case for systems with highly nonlinear dynamics. To solve this problem, we adaptively update $\boldsymbol{\Phi}^N$, $\boldsymbol{\Lambda}$, and $\mathbf{C}$ via procedures discussed in subsection 3.2–subsection 3.4.

Specifically, we set a series of check points in the time span $0 < \tau_1 < \tau_2 < \cdots < \tau_n < T$. On each of the points, the algorithm examines whether the neighborhood of $\boldsymbol{x}(\tau_k)$ is "valid" so as to further guarantee the accuracy of the finite-dimensional approximation. For such a purpose, we define the acceptable range

$$R_i := [L_i + \gamma r, U_i - \gamma r], \tag{3.10}$$

where L_i, U_i are the lower and upper bounds, r is the radius mentioned in subsection 3.4, and $\gamma \in (0, 1]$ is a tunable parameter. At the initial time point, $L_i = x_0^i - r$ and $U_i = x_0^i + r$. For the current state $\boldsymbol{x}(\tau_k) = (x_1(\tau_k), x_2(\tau_k), \ldots, x_d(\tau_k))^\top$, the neighborhood $R_1 \times R_2 \times \cdots \times R_d$ is valid if $x_i(\tau_k) \in R_i$ for all i. In the case where at least one component $x_i(\tau_k) \notin R_i$, we realize the update by the following procedures:

1. Update $L_i = x_i(\tau_k) - r$, $U_i = x_i(\tau_k) + r$ for all i.
2. Generate the sparse grid and compute matrices $\mathbf{M}, \mathbf{G}_i$.
3. Apply the eigendecomposition to update $\boldsymbol{\Phi}^N, \boldsymbol{\Lambda}$.
4. Compute the Koopman modes $\mathbf{C}$ with the updated $\boldsymbol{\Phi}^N$.
5. Construct solution $\boldsymbol{x}(t)$ by replacing $e^{\tilde{\lambda}_j t}$ with $e^{\tilde{\lambda}_j (t - \tau_k)}$ in (3.9).

Step 5 above comes from the adjustment $t_0 = \tau_k$ and $\boldsymbol{x}_0 = \boldsymbol{x}(t_0) = \boldsymbol{x}(\tau_k)$ whenever the update is performed. Notably, the parameter γ controls the strictness of the validity check. When γ is large, the updates occur more frequently. Setting $\gamma = 1$ is tantamount to forcing an update at every check point. As addressed in [24], SASK also differs from traditional ODE solvers as it does not discretize the system in time, and the check points are essentially different from the time grid points in traditional solvers. Instead, the discretization is in the state space. As a result, SASK is time-mesh-independent.

3.6. Algorithm summary. To leverage the properties of the Koopman operator, subsections 3.2–3.4 find the finite-dimensional approximation of the Koopman operator and approximate the eigenfunctions and eigenvalues. The solution is obtained by a linear combination of the eigenfunctions. In order to preserve accuracy as time evolves, the adaptivity is added into the numerical scheme. The complete algorithm is summarized in Algorithm 3.1. Note that the construction of the sparse grid is based on the reference domain $[-1, 1]$ in practice. We then only need to rescale the sparse grid points and matrices $\mathbf{G}_i$ on the reference domain by $\frac{U_i - L_i}{2}(\boldsymbol{\Xi}_i + 1) + L_i$ and $\frac{2\mathbf{G}_i}{U_i - L_i}$, respectively, so that they match the real domain $[L_i, U_i]$.

Algorithm 3.1 Sparse-grid-based adaptive spectral Koopman method.

Require: $n, T, \boldsymbol{x}(0), r, \kappa, \gamma$

1: Set check points at $0 = \tau_0, \tau_1, \ldots, \tau_n < T$.
2: Let $L_i = x_0^i - r_i, U_i = x_0^i + r_i$ and set neighborhood $R_i = [L_i + \gamma r_i, U_i - \gamma r_i]$ for $i = 1, 2, \ldots, d$.
3: Generate sparse grid points $\{\boldsymbol{\xi}_l\}_{l=1}^N$ and compute $\mathbf{M}, \mathbf{G}_i$ for $i = 1, 2, \ldots, d$.
4: Apply eigendecomposition to $\mathbf{UW} = \mathbf{MW\Lambda}$ and compute $\mathbf{\Phi}^N = \mathbf{MW}$.
5: Solve linear system $\mathbf{\Phi}^N \mathbf{C} = \mathbf{\Xi}$, where $\mathbf{\Xi}$ is defined as in subsection 3.4.
6: **for** $k = 1, 2, 3, \ldots, n$ **do**
7: Let ν_j be the first element of the jth column of $\mathbf{\Phi}$. 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 $\mathbf{C}$.
8: **if** $x_i(\tau_k) \notin R_i$ for any i **then**
9: Set $L_i = x_i(\tau_k) - r_i$, $U_i = x_i(\tau_k) + r_i$, and $R_i = [L_i + \gamma r_i, U_i - \gamma r_i]$.
10: Repeat steps 3–5.
11: **end if**
12: **end for**
13: **return** $\boldsymbol{x}(T) = \sum_j \mathbf{C}(j,:)\nu_j e^{\tilde{\lambda}_j(T - \tau_n)}$.

4. Numerical analysis. In this section, we provide numerical analysis results to understand the performance of the ASK and SASK methods. We start with a lemma adapted from [29].

LEMMA 4.1. *Consider a linear dynamical system* $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathbf{A}\boldsymbol{x}$, *where* $\boldsymbol{x} \in \mathbb{R}^d$ *and* $\mathbf{A} \in \mathbb{R}^{d \times d}$. *Then the eigenvalues of* $\mathbf{A}$ *are eigenvalues of the Koopman operator, and the associated Koopman eigenfunctions are* $\varphi_j(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w}_j \rangle$ *for* $j = 1, 2, \ldots, d$, *where* $\boldsymbol{w}_j$ *are eigenvectors of* $\mathbf{A}^\top$ *such that* $\|\boldsymbol{w}_j\|_2 = 1$ *and* $\langle \cdot, \cdot \rangle$ *denotes the complex inner product on the manifold* $\mathcal{M}$.

Proof. Let λ_j be an eigenvalue of $\mathbf{A}$; then it is also an eigenvalue of $\mathbf{A}^\top$. Assume $\mathbf{A}^\top \boldsymbol{w}_j = \lambda_j \boldsymbol{w}_j$, as shown in [29]; we have

$$\frac{\mathrm{d}\varphi_j}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\langle \boldsymbol{x}, \boldsymbol{w}_j \rangle = \left\langle \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}, \boldsymbol{w}_j \right\rangle = \langle \mathbf{A}\boldsymbol{x}, \boldsymbol{w}_j \rangle = \langle \boldsymbol{x}, \mathbf{A}^\top \boldsymbol{w}_j \rangle = \lambda_j \langle \boldsymbol{x}, \boldsymbol{w}_j \rangle = \lambda_j \varphi_j.$$

Thus, φ_j is an eigenfunction of the linear system's Koopman operator. Alternatively, using (3.4), we have

$$\mathcal{K}\varphi_j = \sum_{i=1}^d (\mathbf{A}\boldsymbol{x})_i \frac{\partial \varphi_j}{\partial x_i} = \langle \mathbf{A}\boldsymbol{x}, \boldsymbol{w}_j \rangle = \langle \boldsymbol{x}, \mathbf{A}^\top \boldsymbol{w}_j \rangle = \lambda_j \langle \boldsymbol{x}, \boldsymbol{w}_j \rangle = \lambda_j \varphi_j. \qquad \square$$

These two different proofs of this lemma illustrate the connection of temporal derivatives and spatial derivatives via the Koopman operator.

Next, as long as $\mathbf{A}$ has a full set of eigenvectors at distinct eigenvalues λ_j, we have

$$g(\boldsymbol{x}(t_0 + t)) = \sum_{j=1}^d c_j \varphi_j(\boldsymbol{x}(t_0)) e^{\lambda_j t}$$

for observable $g : \mathcal{M} \to \mathbb{R}$, and $g(\boldsymbol{x}(t_0)) = \sum_{j=1}^d c_j \varphi_j(\boldsymbol{x}(t_0))$. When φ_j and λ_j are perturbed, we have the following convergence estimate.

THEOREM 4.2. *Consider a linear dynamical system* $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathbf{A}\boldsymbol{x}$, *where* $\boldsymbol{x} \in \mathbb{R}^d$ *and* $\mathbf{A} \in \mathbb{R}^{d\times d}$. *Assume* $\mathbf{A}^\top$ *has a full set of eigenvectors at distinct eigenvalues* λ_j. *Let* $\tilde{\varphi}_j(\boldsymbol{x})$ *and* $\tilde{\lambda}_j$ *be approximations of* $\varphi_j(\boldsymbol{x})$ *and* λ_j, *respectively. Consider a scalar observable* $g(\boldsymbol{x}(t_0+t)) = \sum_j^d c_j\varphi_j(\boldsymbol{x}(t_0))e^{\lambda_j t}$. *Denote* $\varepsilon_{\varphi_0} = \max_{1\le j\le d}|\varphi_j(\boldsymbol{x}(t_0)) - \tilde{\varphi}_j(\boldsymbol{x}(t_0))|$ *and* $\varepsilon_\lambda = \max_{1\le j\le d}|\lambda_j - \tilde{\lambda}_j|$. *We have*

$$|g(\boldsymbol{x}(t_0+t)) - \tilde{g}(\boldsymbol{x}(t_0+t))| \le d\|\boldsymbol{c}\|_\infty e^{Re\,\lambda_{\max}t}\left(\max_j|\varphi_j(\boldsymbol{x}(t_0))|\varepsilon_\lambda t + \varepsilon_{\varphi_0}e^{\varepsilon_\lambda t}\right), \tag{4.1}$$

where $\tilde{g}(\boldsymbol{x}(t_0+t)) = \sum_{j=1}^d c_j\tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t}$ *and* $Re\,\lambda_{\max} = \max_j Re\,\lambda_j$.

Proof. With the triangle inequality, we have

$$\begin{aligned}
&|g(\boldsymbol{x}(t_0+t)) - \tilde{g}(\boldsymbol{x}(t_0+t))| \\
&= \left|\sum_j^d c_j\varphi_j(\boldsymbol{x}(t_0))e^{\lambda_j t} - \sum_j^d c_j\tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t}\right| \\
&\le \sum_{j=1}^d |c_j|\cdot\left|\varphi_j(\boldsymbol{x}(t_0))e^{\lambda_j t} - \tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t}\right| \\
&\le \sum_{j=1}^d |c_j|\cdot\Big(\left|\varphi_j(\boldsymbol{x}(t_0))e^{\lambda_j t}\right||1-e^{(\tilde{\lambda}_j-\lambda_j)t}| \\
&\quad + |\varphi_j(\boldsymbol{x}(t_0)) - \tilde{\varphi}_j(\boldsymbol{x}(t_0))|\cdot|e^{\lambda_j t}|\cdot|e^{(\tilde{\lambda}_j-\lambda_j)t}|\Big) \\
&\le d\|\boldsymbol{c}\|_\infty e^{Re\,\lambda_{\max}t}\left(\max_j|\varphi_j(\boldsymbol{x}(t_0))|\cdot|1-e^{\varepsilon_\lambda t}| + \varepsilon_{\varphi_0}e^{\varepsilon_\lambda t}\right).
\end{aligned} \tag{4.2}$$

∎

If the eigendecomposition solver is accurate, then ε_{φ_0} and ε_λ are very close to zero. Consequently, $|1-e^{\varepsilon_\lambda t}| \approx \varepsilon_\lambda t$, and $\tilde{g}$ is very close to g given accurate Koopman modes c_j. For example, when solving linear PDEs, $\mathbf{A}$ can be the differentiation matrix of ∇ or Δ (or other differential operators in the finite difference or pseudospectral method). Hence, it is promising that when solving these PDEs based on their semidiscrete form, the time evolution can be very accurate by ASK. Moreover, since $\varphi_j(\boldsymbol{x}) = \langle\boldsymbol{x}, \boldsymbol{w}_j\rangle$ in the linear case, we can further bound ε_φ and $|\varphi_j(\boldsymbol{x}(t_0))|$ by $\|\boldsymbol{x}(t_0)\|_2$ via the Cauchy–Schwarz inequality.

We note that in Theorem 4.2, Koopman modes c_j are assumed to be accurate. As shown in subsection 3.4, the ASK method computes these modes by solving linear systems based on the computed eigenfunctions $\tilde{\varphi}_j$ using different initial value $g(\boldsymbol{x}(t_0))$. The following theorem provides the error estimate in this practical scenario.

THEOREM 4.3. *Given the condition in Theorem* 4.2, *if the Koopman modes* c_j *are approximated by* $\tilde{c}_j$ *that are computed by solving a linear system* $\tilde{\mathbf{\Phi}}\tilde{\boldsymbol{c}} = \boldsymbol{g}$, *where* $\tilde{\Phi}_{ij} = \tilde{\varphi}_j(\boldsymbol{x}_i)$, $g_j = g(\boldsymbol{x}_j)$, *and* $\boldsymbol{x}_i$ *are* d *different initial values. Let* $\varepsilon_{\varphi_{\max}} = \max_{1\le i\le d}\max_{1\le j\le d}|\varphi_j(\boldsymbol{x}_i) - \tilde{\varphi}_j(\boldsymbol{x}_i)|$. *Consider* $\tilde{g}(\boldsymbol{x}(t_0+t)) = \sum_{j=1}^d \tilde{c}_j\tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t}$, *which is an approximation of* $g(\boldsymbol{x}(t_0+t))$. *Let* $\delta\mathbf{\Phi} = \tilde{\mathbf{\Phi}} - \mathbf{\Phi}$. *If* $\|\mathbf{\Phi}^{-1}\|_\infty\varepsilon_{\varphi_{\max}} < 1/d$, *we have*

$$\begin{aligned}
&|g(\boldsymbol{x}(t_0+t)) - \tilde{g}(\boldsymbol{x}(t_0+t))| \\
&\quad\le d\|\boldsymbol{c}\|_\infty\varepsilon_{\varphi_{\max}}e^{Re\,\lambda_{\max}t}\left(\max_j|\varphi_j(\boldsymbol{x}(t_0))|(|1-e^{\varepsilon_\lambda t}| + \eta e^{\varepsilon_\lambda t}) + \varepsilon_{\varphi_0}e^{\varepsilon_\lambda t}\right),
\end{aligned} \tag{4.3}$$

where $\eta = \frac{\kappa d\varepsilon_{\varphi_{\max}}}{\|\mathbf{\Phi}\|_\infty - \kappa d\varepsilon_\varphi}$ *and* $\kappa = \|\mathbf{\Phi}\|_\infty\cdot\|\mathbf{\Phi}^{-1}\|_\infty$.

Proof. The Koopman modes c_j in this case satisfy $\mathbf{\Phi} \boldsymbol{c} = \boldsymbol{g}$, where $\Phi_{ij} = \varphi_j(\boldsymbol{x}_i)$. Therefore, $\|\delta\mathbf{\Phi}\|_\infty \le d\varepsilon_{\varphi_{\max}}$. Then, a well-known conclusion in numerical linear algebra indicates that

$$\frac{\|\boldsymbol{c} - \tilde{\boldsymbol{c}}\|_\infty}{\|\boldsymbol{c}\|_\infty} \le \frac{\kappa\|\delta\mathbf{\Phi}\|_\infty}{\|\mathbf{\Phi}\|_\infty - \kappa\|\delta\mathbf{\Phi}\|_\infty} \le \frac{\kappa d\varepsilon_{\varphi_{\max}}}{\|\mathbf{\Phi}\|_\infty - \kappa d\varepsilon_{\varphi_{\max}}};$$

we have

$$\begin{aligned}
&|g(\boldsymbol{x}(t_0+t)) - \tilde{g}(\boldsymbol{x}(t_0+t))| \\
&\le \underbrace{\left|\sum_j^d c_j\varphi_j(\boldsymbol{x}(t_0))e^{\lambda_j t} - \sum_j^d c_j\tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t}\right|}_{I} \\
&\quad + \left|\sum_j^d c_j\tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t} - \sum_j^d \tilde{c}_j\tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t}\right| \\
&\le I + \sum_{j=1}^d |c_j - \tilde{c}_j| \cdot \left|\tilde{\varphi}_j(\boldsymbol{x}(t_0))e^{\tilde{\lambda}_j t}\right| \qquad (4.4) \\
&\le I + d\eta\|\boldsymbol{c}\|_\infty \left(\max_j |\varphi_j(\boldsymbol{x}(t_0))| + \varepsilon_{\varphi_{\max}}\right) e^{(Re\,\lambda_{\max}+\varepsilon_\lambda)t} \\
&\le d\|\boldsymbol{c}\|_\infty e^{Re\,\lambda_{\max} t}\left(\max_j |\varphi_j(\boldsymbol{x}(t_0))| \cdot |1 - e^{\varepsilon_\lambda t}| + \varepsilon_{\varphi_0} e^{\varepsilon_\lambda t}\right) \\
&\quad + d\eta\|\boldsymbol{c}\|_\infty \left(\max_j |\varphi_j(\boldsymbol{x}(t_0))| + \varepsilon_{\varphi_{\max}}\right) e^{(Re\,\lambda_{\max}+\varepsilon_\lambda)t} \\
&\le d\|\boldsymbol{c}\|_\infty \varepsilon_{\varphi_{\max}} e^{Re\,\lambda_{\max} t}\left(\max_j |\varphi_j(\boldsymbol{x}(t_0))|(|1 - e^{\varepsilon_\lambda t}| + \eta e^{\varepsilon_\lambda t}) + \varepsilon_{\varphi_0} e^{\varepsilon_\lambda t}\right). \quad \square
\end{aligned}$$

Theorem 4.3 also holds for general cases, i.e., when $\mathbf{f}$ in (2.1) has a full set of eigenvectors at distinct eigenvalues.

Note that these preliminary analysis results provide the first glance on the accuracy of the ASK (as well as the SASK) method, which will serve as the foundation of more comprehensive study. For the most general cases, we need to consider the error of (1) approximating global eigenfunctions locally; (2) approximating local eigenfunctions using predecided basis functions (polynomials in this work); and (3) the eigensolver and the linear solver. Also, we need to investigate the impact of continuous spectrum in some systems, e.g., following the existing works in [1, 9]. These analyses require very systematic study and will be included in our future work. We include an illustrative example in Appendix B to demonstrate the accuracy of approximating desired eigenpairs using SASK for linear equations.

5. Numerical results. SASK's performance is demonstrated by multiple ODEs and PDEs in this section. Specifically, the ODEs are multidimensional nonlinear systems. Our numerical experiments investigate the impact of the parameters in SASK, including approximation level κ, and number of eigendecompositions, denoted by n_d. Note that n_d reflects how adaptivity may increase the accuracy. Additionally, one may refer to [24] for the effect of number of check points n and radius r, for they have an effect on SASK that is similar to their effect on ASK. For ODEs that do not have a closed-form solution, Verner's ninth-order Runge–Kutta (RK9) method [44] is implemented to provide the reference solutions.

To solve a PDE, SASK exploits the semidiscrete form of the PDE, where the spatial discretization is performed by the spectral-collocation method. In this way, the PDE is converted to a high-dimensional ODE system. For such a system, using a large κ will significantly increase the computational cost, so we keep $\kappa = 1$ to investigate the performance of SASK for solving PDEs with this low-order approximation of eigenfunctions.

Regarding the computational efficiency, we perform a comparison between SASK and other state-of-the-art ODE solvers for ODEs examples, including the Euler forward method, the fourth-order Runge–Kutta (RK4) method, and the five-step Adams–Bashforth (AB5) method. Then, we demonstrate the computational efficiency of SASK on the PDEs, where RK4 is incorporated as a comparison to solve the semidiscretized form since it is one of the most widely used methods. The efficiency is measured by the accuracy against the running time (i.e., wall time) in subsection 5.3. All the experiments are implemented in MATLAB 2020 and performed on a machine with Intel Core i7 CPU and 32 GB RAM.

5.1. ODEs. This part summarizes the numerical results of SASK on six nonlinear dynamical systems that are well-known benchmarks across different fields.

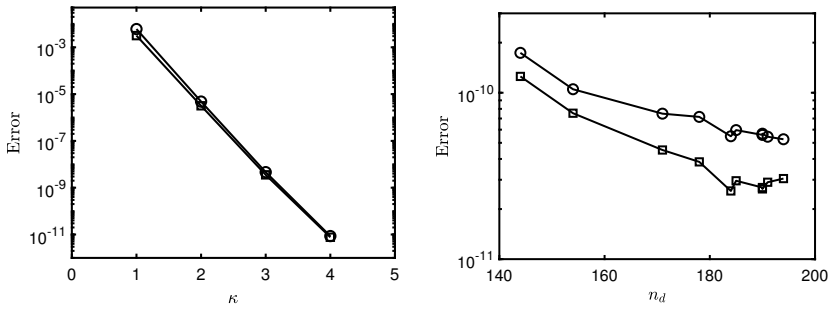
5.1.1. Lotka–Volterra model. The Lotka–Volterra model is a two-dimensional system that describes the interaction between the population evolution of prey and that of predators [5]:

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

The initial state is set to $\boldsymbol{x}(0) = (5, 2)^\top$, and the terminal time is $T = 20$. The two tests corresponding to κ, n_d employ the following specification:

(a) test of κ: $n = 200, r = 0.75, \gamma = 0.8$;
(b) test of n_d: $\kappa = 3, r = 0.5, \gamma = 0.8$.

The results in Figure 1 align with our expectation. First, the error decreases exponentially with the approximation level. This is expected as we leverage the ideas from spectral-collocation methods. On the other hand, adaptivity is shown to contribute to the accuracy, which is illustrated by the downward trending error curves as n_d increases. The small-scale fluctuations come from numerical issues in the eigendecomposition on a small local domain.



(a) Approximation level (b) Number of decompositions

FIG. 1. *Lotka–Volterra model:* $\bigcirc, \square$ *denote* x_1, x_2*, respectively.*

5.1.2. Simple pendulum. In mechanics, the movement of the pendulum can be modeled by an second order ODE, $\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L}\sin(\theta)$, where the gravity acceleration is g, the length of the pendulum is L, and the displacement is θ. Setting $L = g$ for convenience, we can further convert the ODE into an equivalent two-dimensional first order ODE,

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

Here, $x_1 := \theta, x_2 := \frac{\mathrm{d}\theta}{\mathrm{d}t}$. The initial state is $\boldsymbol{x}(0) = \left(-\frac{\pi}{4}, \frac{\pi}{6}\right)^\top$, and the terminal time is $T = 20$. For the simple pendulum, the parameters are listed below:

(a) test of κ: $n = 200, r = 0.2, \gamma = 0.5$;
(b) test of n_d: $\kappa = 3, r = 0.1, \gamma = 0.1$.

As shown in Figure 2, SASK achieves an exponential convergence with respect to κ in this example as well. Also, the error curves flattened when $n_d \geq 160$, indicating that 160 decompositions suffice.

5.1.3. Limit cycle. The limit cycle can be applied to describe the oscillatory patterns [45], which is defined by the two-dimensional ODE as follows:

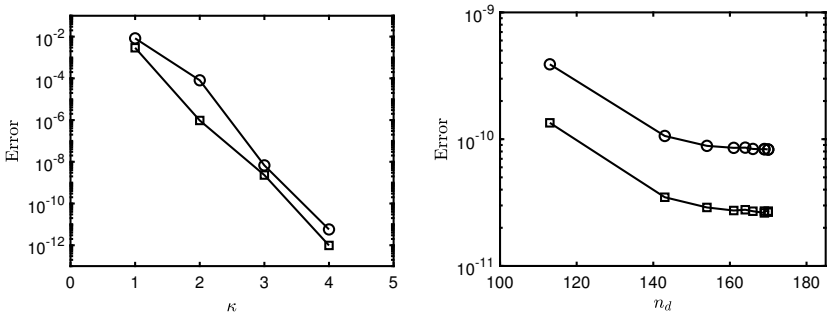
$$\begin{aligned}\frac{\mathrm{d}x_1}{\mathrm{d}t} &= -x_1 - x_2 + \frac{x_1}{\sqrt{x_1^2 + x_2^2}},\\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= x_1 - x_2 + \frac{x_2}{\sqrt{x_1^2 + x_2^2}}.\end{aligned}$$

This model has a closed-form solution,

$$\begin{aligned}x_1(t) &= \left[1 - \left(1 - \sqrt{x_1(0)^2 + x_2(0)^2}\right) e^{-t}\right] \cos(t + \arctan(x_2(0)/x_1(0))),\\ x_2(t) &= \left[1 - \left(1 - \sqrt{x_1(0)^2 + x_2(0)^2}\right) e^{-t}\right] \sin(t + \arctan(x_2(0)/x_1(0))).\end{aligned}$$

The parameters in the experiments are specified as follows:

(a) test of κ: $n = 200, r = \frac{\sqrt{2}}{20}, \gamma = 0.2$;
(b) test of n_d: $\kappa = 3, r = \frac{\sqrt{2}}{40}, \gamma = 0.8$.



(a) Approximation level (b) Number of decompositions

FIG. 2. *Simple pendulum:* ◯,□ *denote* x_1, x_2*, respectively.*

We set $\boldsymbol{x}(0) = (\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})^\top, T = 20$ in the above tests. Similar to the Lotka–Volterra case, Figure 3 demonstrates that the accuracy directly depends on the approximation level. As the states strongly oscillate as time evolves, the validity of the local domain breaks more frequently. Therefore, more decompositions are utilized than the aforementioned models.

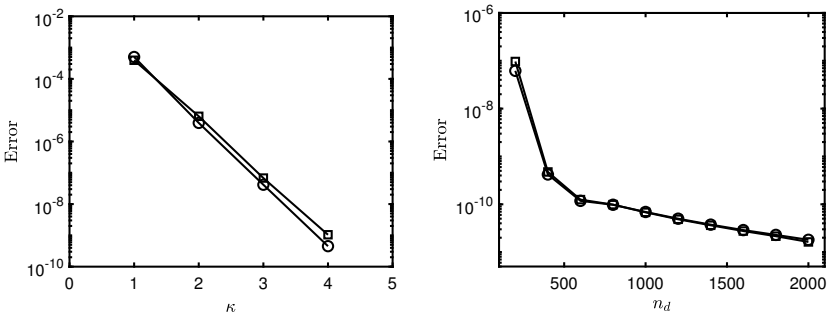
5.1.4. Kraichnan–Orszag model. The Kraichnan–Orszag problem was introduced in [34]. It induces a chaotic three-dimensional dynamical system, defined by the following equations:

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

For this model, $\boldsymbol{x}(0) = (1, 2, -1)^\top, T = 20$. The other parameters are specified as follows:

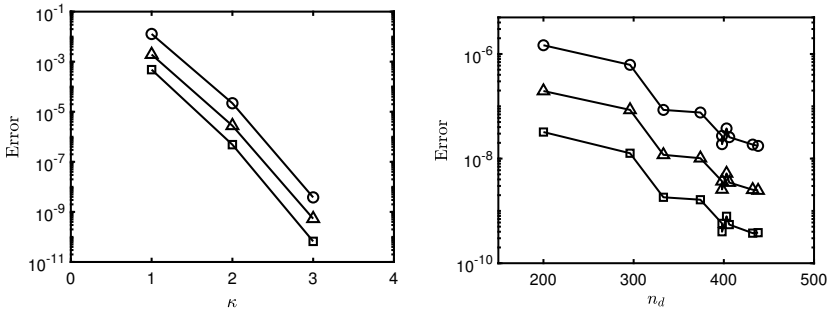
(a) test of κ: $n = 2000, r = 0.5, \gamma = 0.8$;
(b) test of n_d: $\kappa = 3, r = 0.3, \gamma = 0.5$.

The test of κ only includes $\kappa = 1, 2, 3$ because SASK encounters numerical issues when $\kappa = 4$, which leads to a drastic decrease in accuracy. This is because the Kraichnan–Orszag dynamics are less smooth. As illustrated by Figure 4, a large κ as well as a



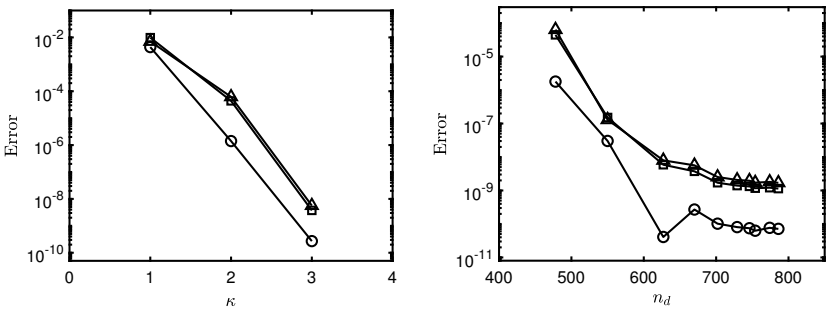
(a) Approximation level (b) Number of decompositions

FIG. 3. *Limit cycle:* ◯,□ *denote* x_1, x_2*, respectively.*



(a) Approximation level (b) Number of decompositions

FIG. 4. *Kraichnan–Orszag model:* ◯,□,△ *denote* x_1, x_2, x_3*, respectively.*



(a) Approximation level (b) Number of decompositions

FIG. 5. *Lorenz attractor:* $\bigcirc, \square, \triangle$ *denote* x_1, x_2, x_3*, respectively.*

large n_d are required to guarantee high accuracy due to the chaotic behavior of the state.

5.1.5. Lorenz attractor. The Lorenz attractor [27] was introduced to model the turbulence in dynamic flows. The system is highly chaotic and exhibits behaviors that pose challenges to solving numerically. The governing equations are shown below:

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

The terminal time is $T = 10$, and the initial state is $\boldsymbol{x}(0) = (3, 2, -1)^\top$. We present the SASK parameters as follows:

(a) test of κ: $n = 2000, r = 1, \gamma = 0.5$;
(b) test of n_d: $\kappa = 3, r = 1, \gamma = 0.5$.

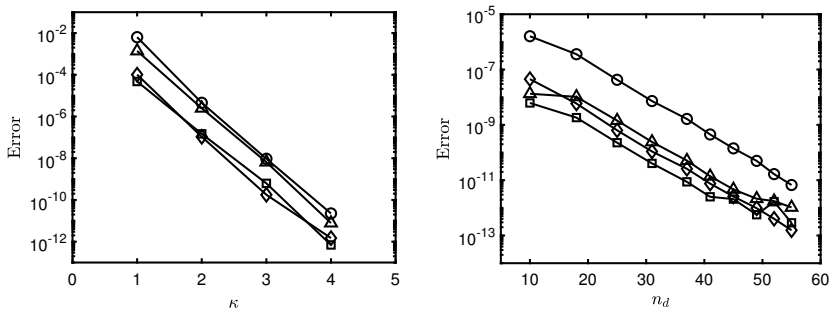
With the chaotic state, SASK needs $\kappa = 3$ to reach a 10^{-8} scale of error. Moreover, Figure 5 shows that n_d has a significant impact on the accuracy until $n_d = 700$, which is not surprising since a local domain becomes invalid quickly.

5.1.6. Influenza virus model. The target cell-limited model with delayed virus production is a four-dimensional dynamical system proposed in [3] to model the kinetics of influenza A virus of individual infection. For convenience, we call it the influenza virus model and write the equations in our notation as follows:

$$\begin{aligned}\frac{\mathrm{d}x_1}{\mathrm{d}t} &= -0.5x_1 x_4,\\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= 0.5x_1 x_4 - 4x_2,\\ \frac{\mathrm{d}x_3}{\mathrm{d}t} &= 4x_2 - 0.5x_3,\\ \frac{\mathrm{d}x_4}{\mathrm{d}t} &= 0.05x_x - 0.5x_4.\end{aligned}$$

For this example, we set $T = 10$ and $\boldsymbol{x}(0) = (4, 2, 2, 1)^\top$. The parameters used in the two experiments are shown below:

(a) test of κ: $n = 1000, r = 0.3, \gamma = 0.2$;
(b) test of n_d: $\kappa = 3, r = 0.3, \gamma = 0.8$.



(a) Approximation level (b) Number of decompositions

FIG. 6. *Influenza virus model:* $\bigcirc, \square, \triangle, \diamond$ *denote* x_1, x_2, x_3, x_4*, respectively.*

Our numerical results on the influenza virus model are illustrated in Figure 6. The exponential convergence manifests in the test of approximation level. Moreover, we do not need many decompositions to achieve a high accuracy, for this model has less chaotic dynamics compared with the Lorenz attractor.

5.2. Solving PDEs. To illustrate SASK's effectiveness, we implemented numerical experiments on four PDEs, the details of which will be exhibited subsequently. Specifically, the spatial discretization is based on the Fourier collocation method (see e.g., [39, 17, 42]) as we impose *periodic* boundary conditions to all the PDEs. The MATLAB code generating differentiation matrices can be found in [42]. Since the induced ODE systems tend to be high-dimensional, we fix the approximation level $\kappa = 1$ so that the computation cost remains feasible as the number of sparse grid points is $2m+1$, if the ODE system is m-dimensional. Suppose the degree of freedom in space is m; then the ODE system is m-dimensional. Nevertheless, the results turn out to be accurate with such a low approximation level. We denote by $y \in \mathbb{R}^m$ the SASK solution at different spatial grid points, and by y^* the exact solution (or the reference solution). The performance of SASK is quantified by the relative L_2 error defined by $\frac{\|y-y^*\|_2}{\|y^*\|_2}$ and L_∞ error defined by $\|y - y^*\|_\infty$.

5.2.1. Advection equation. We consider the following advection equation with an initial condition:

$$\begin{aligned} &\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \quad x \in [0,1], \\ &u(x,0) = 0.2 + \sin(\cos(4\pi x)). \end{aligned} \tag{5.1}$$

The closed-form solution is $u(x,t) = u(x-t,0)$. The collocation points in space are set as $x_j = \frac{j}{32}, j = 0, 1, \ldots, 32$, which leads to a 32-dimensional ODE system. SASK applied the following set of parameters: $n = 1, r = 1, \gamma = 0.2$. The errors at $T = 100$ are $e_{L_2} = 2.41 \times 10^{-11}$ and $e_{L_\infty} = 2.83 \times 10^{-11}$. Figure 7 illustrates the SASK solutions compared with the exact solutions. For the advection equation, SASK only performed eigendecomposition once. The high accuracy is due to the semidiscretized system being linear, and adaptivity was barely triggered.

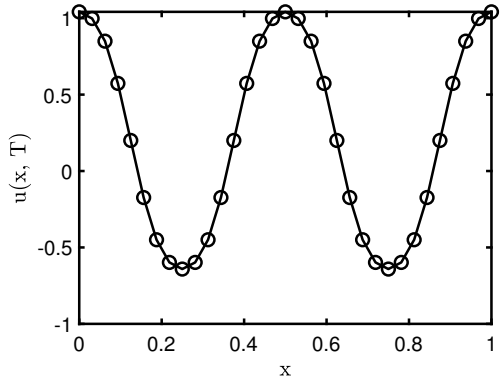


FIG. 7. *Advection equation:* $\bigcirc, -$ *denote SASK and the reference, respectively.*

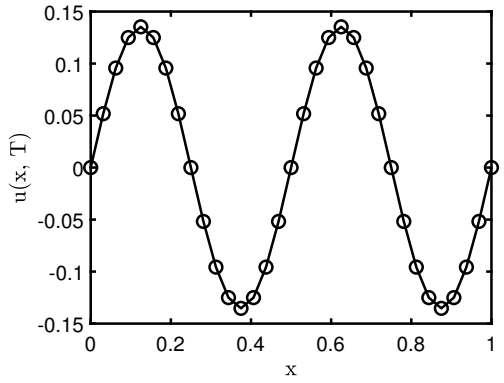


FIG. 8. *Heat equation:* $\bigcirc, -$ *denote SASK and the reference, respectively.*

5.2.2. Heat equation. The following is a heat diffusion equation. Here, we also incorporate an initial condition:

$$\begin{aligned} &\frac{\partial u}{\partial t} = \frac{1}{80\pi^2}\frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1], \\ &u(x,0) = \sin(4\pi x). \end{aligned} \tag{5.2}$$

This equation has a closed-form solution $u(x,t) = \sin(4\pi x)e^{-0.2t}$. Specifying parameters $m = 32, n = 10, r = 1, \gamma = 0.2$, we obtain SASK numerical solutions at $T = 10$. The comparison between the exact solutions with the numerical solutions is exhibited in Figure 8. In particular, $e_{L_2} = 1.98 \times 10^{-14}$ and $e_{L_\infty} = 3.47 \times 10^{-15}$. Also, $n_d = 2$ in this case. Similar to the advection equation case, the semidiscrete form is a linear ODE system, and we observed that $e_{L_2} = 2.29 \times 10^{-14}$ and $e_{L_\infty} = 4.01 \times 10^{-15}$ even when $n_d = n = 1$.

5.2.3. Korteweg–de Vries equation. The next example is the Korteweg–de Vries (KdV) equation with a solitary wave solution:

$$\begin{aligned} &\frac{\partial u}{\partial t} + \beta u\frac{\partial u}{\partial x} + \mu\frac{\partial^3 u}{\partial x^3} = 0, \quad x \in \mathbb{R}, \\ &u(x,0) = \frac{3c}{\beta} sech^2\left(\frac{1}{2}\sqrt{c/\mu}\,(x - (\pi x_0/p + \pi))\right). \end{aligned} \tag{5.3}$$

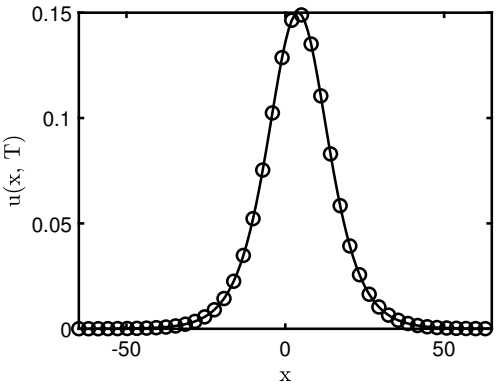


FIG. 9. *KdV equation:* $\bigcirc, -$ *denote SASK and the reference, respectively.*

The closed-form solution is $u(x,t) = \frac{3c}{\beta} sech^2(\frac{1}{2}\sqrt{c/\mu}\,(x - (\pi x_0/p + \pi) - ct))$. Here, β, μ, c are constants, and c is the wave speed. In this test, we set $x_0 = -3.5, c = 0.3, \beta = 6, \mu = 12$. In general, the solution decays to zeros for $|x| >> 1$. Therefore, numerically we solve this equation in a finite domain $[-p, p]$ with a periodic boundary condition. In this example we set $p = 65$. Furthermore, as shown in [13, 39], a change of variable step ($x \to \pi x/p + \pi$) transforms the solution interval from $[-p, p]$ to $[0, 2\pi]$. Consequently, β and μ in (5.3) are replaced by $\tilde{\beta} = \frac{\beta\pi}{p}$ and $\tilde{\mu} = \frac{\mu\pi^3}{p^3}$, respectively. For the spatial discretization, we set $m = 128$ since the behavior of the KdV equation requires finer spatial grids. The SASK parameters are $n = 6, r = 0.1, \gamma = 1$. Since $\gamma = 1$, $n_d = n = 6$. SASK computed the solutions at $T = 25$, which is illustrated in Figure 9. The errors are $e_{L_2} = 1.57 \times 10^{-4}$ and $e_{L_\infty} = 2.51 \times 10^{-5}$.

The KdV equation exhibits more complicated behaviors with two solitons. Therefore, we consider two solitons with speed c_1 and c_2, and $c_1 > c_2$. The closed-form solution of two-soliton KdV can be written as

$$u(x,t) = \frac{12(c_1 - c_2)}{\beta} \frac{s_1^2 \cosh^2(0.5 s_2 z_2) + s_2^2 \sinh^2(0.5 s_1 z_1)}{[(s_1 - s_2)\cosh(0.5(s_1 z_1 + s_2 z_2)) + (s_1 + s_2)\cosh(0.5(s_1 z_1 - s_2 z_2))]^2},$$

where $s_1 = \sqrt{c_1/\mu}, s_2 = \sqrt{c_2/\mu}$, and $z_1 = x - (\pi x_{01}/p + \pi) - c_1 t, z_2 = x - (\pi x_{02}/p + \pi) - c_2 t$. For this example, $x_{01} = -7.5, x_{02} = -5, c_1 = 1, c_2 = 0.5, \beta = 6, \mu = 1, p = 25$. The spatial grid remains with $m = 128$. We set $n = 300, r = 0.1, \gamma = 1$ for SASK and compute the solutions at $T = 15$. The errors are $e_{L_2} = 3.68 \times 10^{-6}$ and $e_{L_\infty} = 1.43 \times 10^{-6}$. Finally, Figure 10 visualizes the solutions.

5.2.4. Burgers equation. The last example is a viscous Burgers equation with an initial condition considered:

$$\begin{aligned} &\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1], \\ &u(x,0) = 0.2 + \sin(2\pi x). \end{aligned} \tag{5.4}$$

The advection term uu_x is treated in the conservation form, i.e., $\frac{1}{2}(u^2)_x$. Particularly, the reference solution is obtained by the high-order spectral method. For this example, the degree of freedom in space is $m = 64$, and SASK admits the parameters $n = 100, r = 0.1, \gamma = 1$. Here, $n_d = n = 100$ because $\gamma = 1$. With $\nu = 0.005$ and $T = 1$,

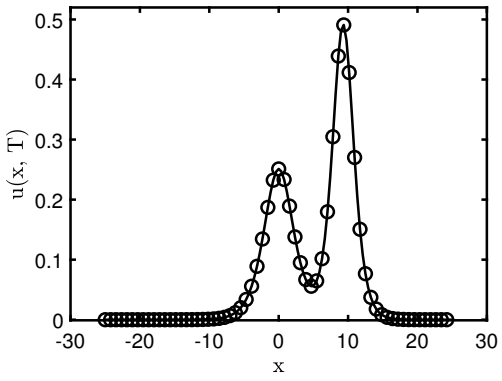


FIG. 10. *Two-soliton KdV equation:* $\bigcirc, -$ *denote SASK and the reference, respectively.*

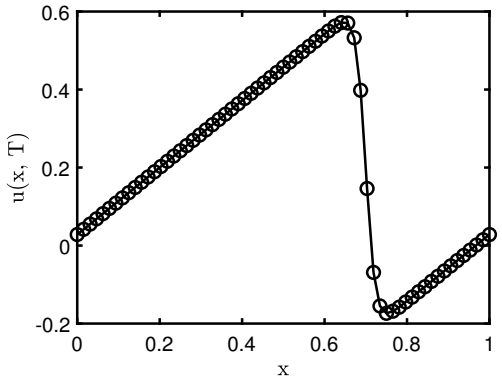


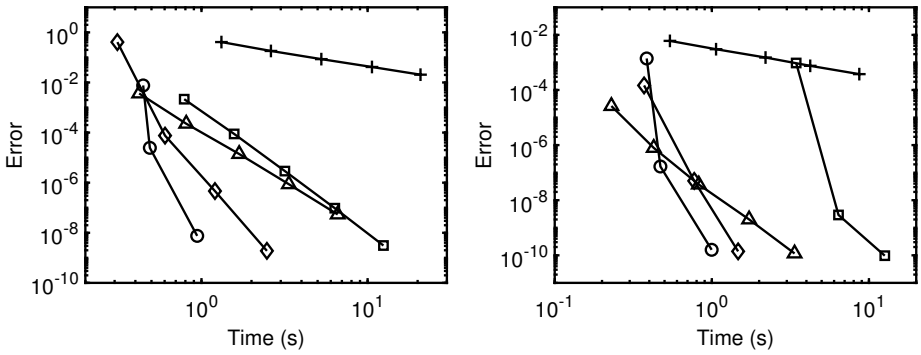
FIG. 11. *Burgers equation:* $\bigcirc, -$ *denote SASK and the reference, respectively.*

SASK yields $e_{L_2} = 4.94 \times 10^{-4}$ and $e_{L_\infty} = 6.12 \times 10^{-4}$. The result is presented in Figure 11, which indicates that the solution is accurate before the discontinuity is fully developed. Additionally, we will observe the Gibbs phenomenon near the discontinuity if we keep increasing T.

Remark 5.1. We test two linear PDEs and two nonlinear PDEs. The purpose of starting with linear PDEs is to demonstrate that (1) ASK (or SASK) is different from conventional ODE solvers and it does not have a restriction like the CFL condition or drawbacks like numerical viscosity; (2) the Koopman operator's eigenpairs can be accurately approximated in the linear cases, since the solution by SASK remains very accurate even without the adaptivity for a large T, although SASK computes the eigenpairs differently from the closed-form formulae shown in Lemma 4.1.

5.3. Efficiency comparison.

5.3.1. Comparison on ODEs. Following a similar idea from the efficiency test in [24], we incorporate numerical tests that focus on the accuracy against the running time, and we compare SASK with conventional ODE solvers. The accuracy is measured by the error defined as $\sqrt{\frac{\sum_i^d e_i^2}{d}}$, where e_i is the integration error in dimension i. More importantly, to demonstrate the impact of the cost of evaluating $\mathbf{f}$ we artificially slow down its evaluation in the codes. Specifically, each $\mathbf{f}$ evaluation is repeated $10,000$ times in both the Kraichnan–Orszag model and the influenza virus



(a) Kraichnan-Orszag model (b) Influenza virus model

FIG. 12. *Comparison of computational efficiency: Error against running time. Symbols* $\bigcirc, +, \triangle, \diamond, \square$ *denote SASK, Euler, RK*4, *RK*9, *and AB*5, *respectively. The time of evaluating* $\mathbf{f}$ *is artificially increased in the code.*

model before the output is returned in the codes. Under this circumstance, SASK gains an advantage over the conventional numerical solvers in the Kraichnan–Orszag model and the influenza virus model.

Herein, SASK starts with $\kappa = 1$, which then increases to 2 and 3. The time window is fixed at $[0, 10]$ in the experiments. For the Kraichnan–Orszag model with $\boldsymbol{x}(0) = (1, 2, -1)^\top$, SASK has $n = 200, r = 0.5, \gamma = 0.8$, while $n = 1000, r = 0.2, \gamma = 0.2$ in the influenza virus model with $\boldsymbol{x}(0) = (4, 2, 2, 1)^\top$. As for the conventional solvers, we begin with a large time step and keep reducing it by half.

Figure 12 demonstrates that SASK is more efficient than the conventional solvers when high accuracy is desired. As expected, RK9 is the most comparable competitor among the state-of-the-art schemes. Additionally, SASK exhibits an "elbow-shaped" pattern when κ rises from 2 to 3. This is a numerical issue induced by the increase of condition numbers of matrices used in the eigendecomposition.

5.3.2. Comparison on PDEs. To demonstrate the efficiency of SASK compared with RK4, we focus on the running time of the two methods based on the PDEs. For RK4, the time step size is set as follows:

(1) advection equation: $\Delta t = 0.01\Delta x = 3.13 \times 10^{-4}$;
(2) heat equation: $\Delta t = 0.9(\Delta x)^2 = 8.79 \times 10_{-4}$;
(3) KdV equation: $\Delta t = 0.006(\Delta x)^3 = 6.30 \times 10^{-3}$;
(4) two-soliton KdV equation: $\Delta t = 0.09(\Delta x)^3 = 5.40 \times 10^{-3}$;
(5) Burgers equation: $\Delta t = 0.5(\Delta x)^2 = 1.22 \times 10^{-4}$.

Here, we use a small Δt for the advection equation to improve the accuracy since T is large. On the other hand, SASK parameters remain the same as in subsection 5.2. Also, similar to the comparison experiments on ODEs, we artificially slow down the dynamics evaluation in the two-soliton KdV example by repeating the evaluation 100 times to mimic the practical case with nonlinearity and high computational cost for evaluating the right-hand side.

We summarize the results in Table 1. The first row of each PDE is the running time, while the second and the third are the relative L_2 error and the L_∞ error. For the advection equation and the heat equation, the dynamics of their semidiscrete form are linear, so SASK needs only 1 or 2 decompositions to be accurate. This results in SASK's higher computational efficiency. In contrast, the Burgers equation

TABLE 1
Performance comparison between SASK and RK4 on PDEs (the dynamics evaluation in two-soliton KdV is artificially slowed down in the code).

		SASK	RK4
Advection equation	time (s)	0.0036	2.0867
	e_{L_2}	2.48e-11	2.82e-08
	eL_∞	2.83e-11	3.91e-08
Heat equation	time (s)	0.0085	0.0829
	e_{L_2}	1.98e-14	4.41e-13
	e_{L_∞}	3.47e-15	6.05e-14
KdV equation	time (s)	0.4194	0.5508
	e_{L_2}	1.57e-04	1.26e-04
	e_{L_∞}	2.51e-05	2.81e-05
two-soliton KdV equation	time (s)	26.2199	39.9604
	e_{L_2}	3.68e-06	3.41e-06
	e_{L_∞}	1.43e-06	1.43e-06
Burgers equation	time (s)	1.0962	0.2250
	e_{L_2}	4.94e-04	4.88e-04
	e_{L_∞}	6.13e-04	4.96e-04

and KdV equation cost SASK more decompositions to preserve the accuracy since the dynamics are nonlinear. While SASK has a comparable performance to RK4 for the KdV equation, it needs around 5 times the time of RK4 to reach a comparable accuracy for the Burgers equation. This is because SASK requires more adaptivity steps to accurately track the evolution of the states in the Burgers equation, which is probably due to the regularity of the system itself. The two-soliton KdV test indicates that SASK can be more efficient than state-of-the-art methods when evaluating the dynamics is costly even though it requires more eigendecomposition steps for nonlinear problems. This is because it enables evaluating the dynamics in parallel.

6. Conclusion and discussion. In this work, we propose the sparse-grid-based ASK method to solve autonomous dynamical systems. Leveraging the sparse grid method, SASK is an efficient extension of the ASK for (high-dimensional) ODE systems. Also, we demonstrate SASK's potential to solve PDEs efficiently by solving semidiscrete systems. In particular, our numerical results demonstrate that if the semidiscrete form is a linear function of the discretized solution of the PDE, SASK is very accurate and much more efficient than conventional ODE-solver-based methods when it is costly to evaluate the dynamics. Furthermore, by selecting the adaptivity criteria carefully, SASK can solve highly nonlinear PDEs like the KdV equation and deal with large total variation in the solution like the Burgers equation.

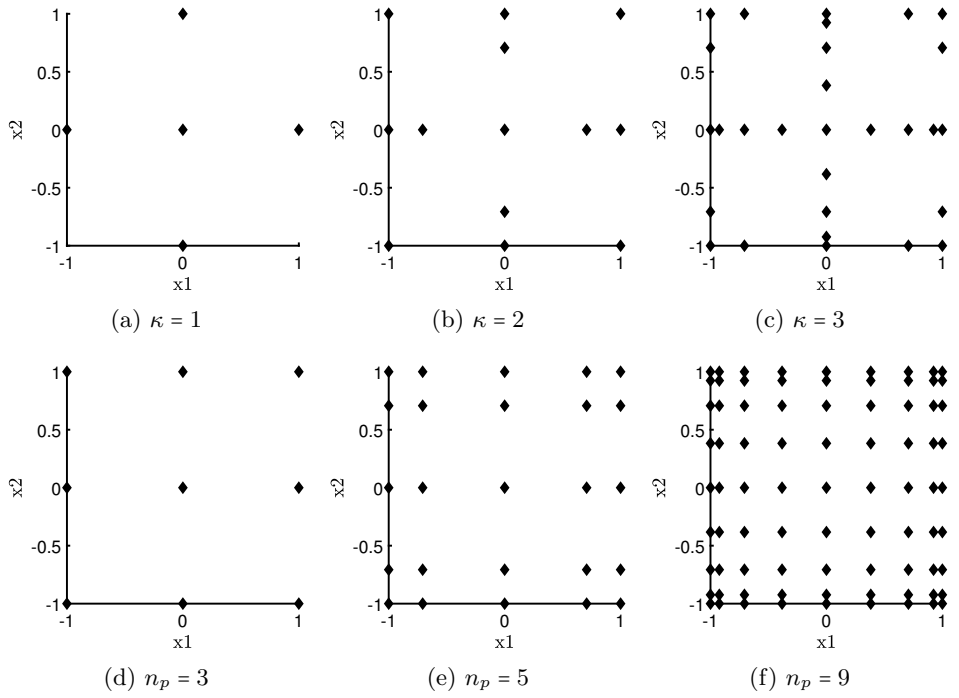
Regarding the sampling strategy, we test multiple level κ for ODEs but fix $\kappa = 1$ in the PDE tests to keep a low computational cost (and SASK still obtains good results in the illustrative examples). In our future work, we will further investigate the selection of adaptivity parameters and the requirement on the accuracy level of the sparse grid method. For example, different sparse grid structures may result in different convergence rates. Also, anisotropic or adaptive sparse grids [7] may enhance the accuracy with a minimal increase in the computational cost.

Finally, it is possible to use randomized algorithms [32] combined with different basis functions like radial basis functions or activation functions used in neural networks to further enhance the accuracy and efficiency of ASK for some high-dimensional problems. Due to the close connection between data-driven methods like DMD (and

its variants) and the Koopman operator, new developments on these methods may be applicable to ASK (and SASK). For example, randomized algorithms have shown great success in enhancing the efficiency in DMD [11]. Also, various numerical methods can be used to approximate eigenfunctions such as radial basis functions in a reproducing kernel Hilbert space [8] and neural networks [25, 26], etc. For PDEs, it is possible to directly approximate the Koopman eigenfunctions without implementing a spatial discretization for the equation first [4]. More discussions can be found in [46, 31, 30, 15]. These works can motivate various ways of improving the design of ASK/SASK.

Appendix A. Sparse grid illustration. Figure 13 is an illustration of sparse grids and their full grid counterparts. To give a brief view of the growth in the grid size, Table 2 shows that the full grid grows much faster than the sparse grid. Notably, the sparse grid grows slowly.

Appendix B. Approximation of eigenpairs in a linear system. We present an illustrative example to investigate the accuracy of approximating eigenpairs for linear systems. We consider the following system:



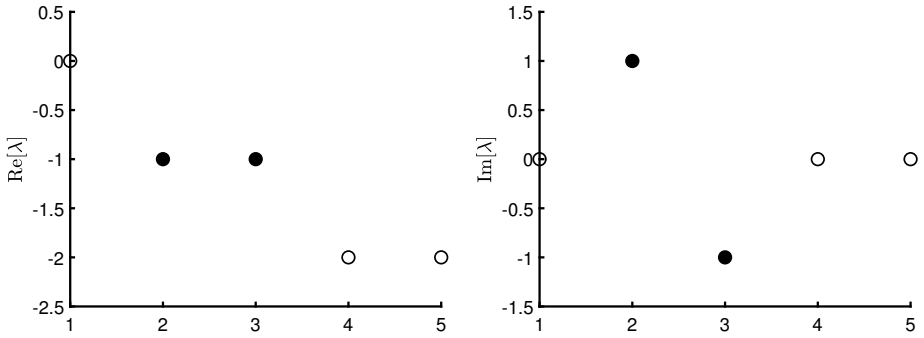
(a) $\kappa = 1$ (b) $\kappa = 2$ (c) $\kappa = 3$

(d) $n_p = 3$ (e) $n_p = 5$ (f) $n_p = 9$

FIG. 13. *Sparse grid (first row); full grid (second row): n_p denotes the number of points in each dimension.*

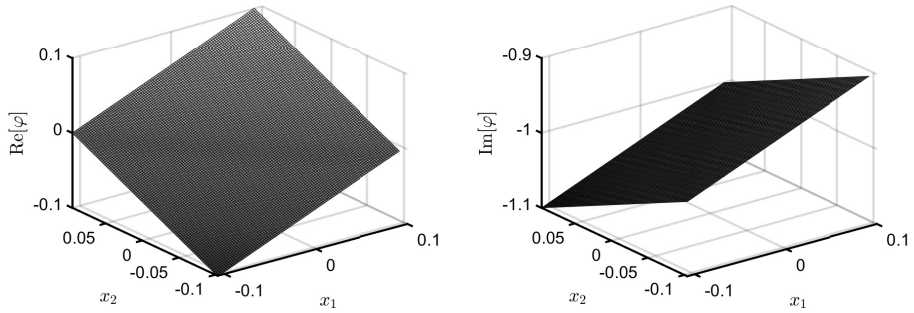
TABLE 2
Grid growth: n_p denotes the number of points in each dimension.

d	$\kappa = 1$	$n_p = 3$	$\kappa = 2$	$n_p = 5$	$\kappa = 3$	$n_p = 9$
2	5	9	13	25	29	81
3	7	27	25	125	69	729
4	9	81	41	625	137	6561
5	11	243	61	3125	241	59049



(a) Real part (b) Imaginary part

FIG. 14. *Koopman eigenvalues of 2D linear ODE: The desired eigenvalues are marked by solid points.*



(a) Real part (b) Imaginary part

FIG. 15. *Koopman eigenfunction of 2D linear ODE.*

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathbf{A}\boldsymbol{x}, \text{ with } \mathbf{A} = \begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix},$$

and $\boldsymbol{x}(0) = (-1, 1)^\top, T = 5$. Notably, we conduct eigendecomposition at $t = 0$ with $\kappa = 1$ to approximate eigenvalues and construct eigenfunctions. The total number of collocation points is 5, so, numerically, we have 5 eigenpairs. We then compute the Koopman modes and find that only 2 out of 5 are nonzero in the numerical computation in SASK. On the other hand, the closed-form formula of the Koopman eigenpairs shown in Lemma 4.1 indicates that there are only 2 eigenpairs. Specifically, the two eigenvalues of $\mathbf{A}$ are $\lambda_1 = -1 + i, \lambda_2 = -1 - i$. Figure 14 plots the real and imaginary parts of the 5 eigenvalues obtained in SASK, and the two eigenvalues corresponding to nonzeros Koopman modes are marked as solid points. They are exactly the desired ones according to Lemma 4.1. The accuracy of approximating these two eigenvalues is at the level of 10^{-14}. Next, as we construct eigenfunctions based on SASK in a small neighborhood of $\boldsymbol{x}(0)$ numerically (see Figure 15), we evaluate these functions at $t = 5$ (i.e., based on $\boldsymbol{x}(5)$), and no additional decomposition is used. The parameters are $\kappa = 1, r = 0.1$. Apparently, at $t = 5$, $\boldsymbol{x}$ has moved out of the neighborhood of $\boldsymbol{x}(0)$. The relative L_2 error is at the level of 10^{-15}. This example illustrates that even though our method computes eigenpairs in a different way from the theoretical approach and yields more eigenpairs than desired, it can

accurately approximate desired eigenpairs. And, due to the linearity of eigenfunctions, extrapolation of the numerically computed eigenfunctions is still accurate.

REFERENCES

[1] H. Arbabi and I. Mezić, *Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator*, SIAM J. Appl. Dyn. Syst., 16 (2017), pp. 2096–2126, https://doi.org/10.1137/17M1125236.

[2] T. Askham and J. N. Kutz, *Variable projection methods for an optimized dynamic mode decomposition*, SIAM J. Appl. Dyn. Syst., 17 (2018), pp. 380–416, https://doi.org/10.1137/M1124176.

[3] P. Baccam, C. Beauchemin, C. A. Macken, F. G. Hayden, and A. S. Perelson, *Kinetics of influenza A virus infection in humans*, J. Virol., 80 (2006), pp. 7590–7599.

[4] M. Balabane, M. A. Mendez, and S. Najem, *Koopman operator for Burgers' equation*, Phys. Rev. Fluids, 6 (2021), 064401.

[5] I. M. Bomze, *Lotka-Volterra equation and replicator dynamics: A two-dimensional classification*, Biol. Cybernet., 48 (1983), pp. 201–211.

[6] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, *Modern Koopman Theory for Dynamical Systems*, preprint, arXiv:2102.12086, 2021.

[7] H.-J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269.

[8] S. Das and D. Giannakis, *Koopman spectra in reproducing kernel Hilbert spaces*, Appl. Comput. Harmon. Anal., 49 (2020), pp. 573–607.

[9] F. Dietrich, T. N. Thiem, and I. G. Kevrekidis, *On the Koopman operator of algorithms*, SIAM J. Appl. Dyn. Syst., 19 (2020), pp. 860–885, https://doi.org/10.1137/19M1277059.

[10] A. S. Dogra and W. Redman, *Optimizing neural networks via Koopman operator theory*, in Advances in Neural Information Processing Systems 33, Curran Associates, 2020, pp. 2087–2097.

[11] N. B. Erichson, L. Mathelin, J. N. Kutz, and S. L. Brunton, *Randomized dynamic mode decomposition*, SIAM J. Appl. Dyn. Syst., 18 (2019), pp. 1867–1891, https://doi.org/10.1137/18M1215013.

[12] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1998.

[13] B. Fornberg and G. B. Whitham, *A numerical and theoretical study of certain nonlinear wave phenomena*, Philos. Trans. Roy. Soc. London Ser. A Math. Phys. Sci., 289 (1978), pp. 373–404.

[14] T. Gerstner and M. Griebel, *Numerical integration using sparse grids*, Numer. Algorithms, 18 (1998), pp. 209–232.

[15] D. Giannakis, *Data-driven spectral decomposition and forecasting of ergodic dynamical systems*, Appl. Comput. Harmon. Anal., 47 (2019), pp. 338–396.

[16] M. Griebel, M. Schneider, and C. Zenger, *A combination technique for the solution of sparse grid problems*, in Iterative Methods in Linear Algebra (Brussels, 1991), North-Holland, Amsterdam, 1992, pp. 263–281.

[17] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb, *Spectral Methods for Time-Dependent Problems*, Cambridge Monogr. Appl. Comput. Math. 21, Cambridge University Press, 2007.

[18] K. L. Judd, L. Maliar, S. Maliar, and R. Valero, *Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain*, J. Econom. Dynam. Control, 44 (2014), pp. 92–123.

[19] B. O. Koopman, *Hamiltonian systems and transformation in Hilbert space*, Proc. Natl. Acad. Sci. USA, 17 (1931), 315.

[20] M. Korda, M. Putinar, and I. Mezić, *Data-driven spectral analysis of the Koopman operator*, Appl. Comput. Harmon. Anal., 48 (2020), pp. 599–629.

[21] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, SIAM, Philadelphia, 2016, https://doi.org/10.1137/1.9781611974508.

[22] J. N. Kutz, X. Fu, and S. L. Brunton, *Multiresolution dynamic mode decomposition*, SIAM J. Appl. Dyn. Syst., 15 (2016), pp. 713–735, https://doi.org/10.1137/15M1023543.

[23] J. N. 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), 6010634.

[24] B. Li, Y. Ma, J. N. Kutz, and X. Yang, *The adaptive spectral Koopman method for dynamical systems*, SIAM J. Appl. Dyn. Syst., 22 (2023), pp. 1523–1551, https://doi.org/10.1137/22M1487941.
[25] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis, *Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator*, Chaos, 27 (2017), 103111.
[26] Y. Liu, A. Sholokhov, H. Mansour, and S. Nabi, *Physics-Informed Koopman Network*, preprint, arXiv:2211.09419, 2022.
[27] E. N. Lorenz, *Deterministic nonperiodic flow*, J. Atmos. Sci., 20 (1963), pp. 130–141.
[28] I. Mezić, *Spectral properties of dynamical systems, model reduction and decompositions*, Nonlinear Dyn., 41 (2005), pp. 309–325.
[29] I. Mezić, *Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry*, J. Nonlinear Sci., 30 (2020), pp. 2091–2145.
[30] I. Mezić, *On numerical approximations of the Koopman operator*, Mathematics, 10 (2022), 1180.
[31] H. Nakao and I. Mezić, *Spectral analysis of the Koopman operator for partial differential equations*, Chaos, 30 (2020), 113131.
[32] Y. Nakatsukasa and J. A. Tropp, *Fast and accurate randomized algorithms for linear systems and eigenvalue problems*, SIAM J. Matrix Anal. Appl., 45 (2024), pp. 1183–1214.
[33] M. Netto and L. Mili, *A robust data-driven Koopman Kalman filter for power systems dynamic state estimation*, IEEE Trans. Power Syst., 33 (2018), pp. 7228–7237.
[34] S. A. Orszag and L. Bissonnette, *Dynamical properties of truncated Wiener-Hermite expansions*, Phys. Fluids, 10 (1967), pp. 2603–2613.
[35] J. Page and R. R. Kerswell, *Koopman analysis of Burgers equation*, Phys. Rev. Fluids, 3 (2018), 071901.
[36] J. L. Proctor, S. L. Brunton, and J. N. Kutz, *Dynamic mode decomposition with control*, SIAM J. Appl. Dyn. Syst., 15 (2016), pp. 142–161, https://doi.org/10.1137/15M1013857.
[37] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, *Spectral analysis of nonlinear flows*, J. Fluid Mech., 641 (2009), pp. 115–127.
[38] P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, J. Fluid Mech., 656 (2010), pp. 5–28.
[39] J. Shen and T. Tang, *Spectral and High-Order Methods with Applications*, Science Press, Beijing, 2006.
[40] S. A. Smolyak, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, in Doklady Akademii Nauk, Vol. 148, Russian Academy of Sciences, 1963, pp. 1042–1045.
[41] A. Surana and A. Banaszuk, *Linear observer synthesis for nonlinear systems using Koopman operator framework*, IFAC-PapersOnLine, 49 (2016), pp. 716–723.
[42] L. N. Trefethen, *Spectral Methods in MATLAB*, Software Environ. Tools 10, SIAM, Philadelphia, 2000, https://doi.org/10.1137/1.9780898719598.
[43] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, *On dynamic mode decomposition: Theory and applications*, J. Comput. Dyn., 1 (2014), pp. 391–421.
[44] J. H. Verner, *Explicit Runge–Kutta methods with estimates of the local truncation error*, SIAM J. Numer. Anal., 15 (1978), pp. 772–790, https://doi.org/10.1137/0715051.
[45] M. Vidyasagar, *Nonlinear Systems Analysis*, Classics in Appl. Math. 42, SIAM, Philadelphia, 2002, https://doi.org/10.1137/1.9780898719185.
[46] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition*, J. Nonlinear Sci., 25 (2015), pp. 1307–1346.
[47] D. Wilson and J. Moehlis, *Isostable reduction with applications to time-dependent partial differential equations*, Phys. Rev. E, 94 (2016), 012211.
[48] C. Zenger and W. Hackbusch, *Sparse grids*, in Proceedings of the Research Workshop of the Israel Science Foundation on Multiscale Phenomena, Modelling and Computation, 1991, 86.