Industrial and
Systems Engineering

The Sparse-Grid-Based Adaptive Spectral Koopman Method

BIAN LI[1], YUE YU[2], AND XIU YANG[1]

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

[2]Department of Mathematics, Lehigh University, Bethlehem, PA, USA

ISE Technical Report 23T-009

LEHIGH UNIVERSITY

The Sparse-Grid-Based Adaptive Spectral Koopman Method

Bian Li[*] Yue Yu[†] Xiu Yang[*]

January 20, 2023

Abstract

The adaptive spectral Koopman (ASK) method was introduced to numerically solve autonomous dynamical systems that lay the foundation of numerous applications across different fields in science and engineering. Although ASK achieves high accuracy, it is computationally more expensive for multi-dimensional 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 multi-dimensional systems. This sparse-grid-based ASK (SASK) method uses the Smolyak structure to construct multi-dimensional 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. Numerical experiments illustrate that SASK balances the accuracy with the computational cost, and hence accelerates ASK. Moreover, we demonstrate that SASK can be used to solve partial differential equations based-on their semi-discrete form.

Keywords: *dynamical systems, sparse grids, Koopman operator, spectral-collocation method, partial differential equations*

1 Introduction

The Koopman operator [12] 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 use them to represent dynamically interpretable low-dimensional embeddings of high-dimensional state spaces to construct solutions through linear superposition [3]. 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 [19, 13, 20].

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

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

[†]Department of Mathematics, Lehigh University, Bethlehem, PA (yuy214@lehigh.edu)

on the subspace spanned by the chosen observables [18]. This approach has been applied to study ordinary differential equations (ODEs), partial differential equations (PDEs) [36, 21, 24, 20], disspative dynamical systems [19], 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) [26, 27, 32, 25, 15, 21, 1] and its variants like extended DMD (EDMD) [35] 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 designing filters (e.g., [30, 22]), training neural networks (e.g., [5]), etc.

In [16] we propose a novel numerical method based on the spectral-collocation method (i.e., the pseudospectral method) [6, 31] to implement the Koopman operator approach to solving nonlinear ordinary differential equations. This method leverages the differentiation matrix in spectral methods to approximate the generator of the Koopman operator, and then conducts eigen-decomposition 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 Koopman modes are approximated by computing eigenvalues, eigenvectors, and the observable based on the initial state. This approach is more efficient than the conventional Runge-Kutta scheme for one-dimensional system as shown in [16]. However, its efficiency decreases as the system's dimension increases as it employs the tensor product rule to construct multi-dimensional 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 eigen-decomposition problem and the linear system in the ASK scheme. Therefore, ASK is less efficient in multi-dimensional cases. To overcome this difficulty, we propose to combine the sparse grids method with ASK, wherein the Smolyak structure is applied to construct collocations points. This sparse-grids-based ASK (SASK) method reduces the number of collocation points and of basis functions used in the vanilla ASK. Hence, the computational efficiency is enhanced. Moreover, we demonstrate that SASK can also solve (nonlinear) PDEs accurately based on their semi-discrete form.

The paper is organized as follows. Background topics are introduced in section 2. Then, the sparse-grid-based 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 [14], we consider an autonomous system described by the ordinary differential equations

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \mathbf{f}(\boldsymbol{x}), \tag{1}$$

where the state $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^\top$ belongs to an 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 [18]).

The flow map $\mathbf{F}_t : \mathscr{M} \to \mathscr{M}$ induced by the dynamical system (1) depicts the evolution of the system as

$$\boldsymbol{x}(t_0 + t) = \mathbf{F}_t(\boldsymbol{x}(t_0)) = \boldsymbol{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\boldsymbol{x}(s))\, ds. \tag{2}$$

Now we define the Koopman operator for continuous-time dynamical systems as follows [19]:

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 (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 [18, 26] 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 brievity) of the Koopman operator. Specifically, for any smooth obervable function g, the generator of the Koopman operator $\mathcal{K}_t$, denoted as $\mathcal{K}$, is given by

$$\mathcal{K} g = \lim_{t\to 0} \frac{\mathcal{K}_t g - g}{t}, \tag{3}$$

which leads to

$$\mathcal{K} g(\boldsymbol{x}) = \nabla g(\boldsymbol{x}) \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\mathrm{d}g(\boldsymbol{x})}{\mathrm{d}t}. \tag{4}$$

Denoting φ an eigenfunction of $\mathcal{K}$ and λ the eigenvalue associated with φ, we have $\mathcal{K}\varphi(\boldsymbol{x}) = \lambda\varphi(\boldsymbol{x})$, , 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.,

$$\mathcal{K}_t\varphi(\boldsymbol{x}(t_0)) = e^{\lambda t}\varphi(\boldsymbol{x}(t_0)). \tag{5}$$

Therefor, φ is an eigenfunction of $\mathcal{K}_t$ associated with eigenvalue λ. Of note, following notations in literatur, 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

$$\mathcal{K}_t[g(\boldsymbol{x}(t_0))] = \mathcal{K}_t\left[\sum_j^\infty c_j\varphi_j(\boldsymbol{x}(t_0))\right] = \sum_j^\infty c_j\mathcal{K}_t[\varphi_j(\boldsymbol{x}(t_0))]. \tag{6}$$

Hence,

$$g(\boldsymbol{x}(t_0 + t)) = \sum_j^\infty c_j\varphi_j(\boldsymbol{x}(t_0))e^{\lambda_j t}. \tag{7}$$

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

$$\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x})}{\mathrm{d}t} = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{K}g_1(\boldsymbol{x}) \\ \mathcal{K}g_2(\boldsymbol{x}) \\ \vdots \\ \mathcal{K}g_d(\boldsymbol{x}) \end{bmatrix} = \sum_j^{\infty} \lambda_j \varphi_j(\boldsymbol{x}) \boldsymbol{c}_j, \tag{8}$$

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

The ASK method uses the following truncated form of Eq (7)

$$g(\boldsymbol{x}(t_0 + t)) = \sum_j^{\infty} c_j \varphi_j(\boldsymbol{x}(t_0)) e^{\lambda_j t} \approx \sum_{j=0}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) e^{\tilde{\lambda}_j t} \tag{9}$$

for $d = 1$. Here, φ_j are approximated using N-th 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 $\tilde{c}_j$ [16]. For $d > 1$, φ_j^N is constructed by tensor product rule using one-dimensional interpolatoion polynomials.

3 Sparse-Grid-Based Adaptive Spectral Koopman Method

As mentioned in [16], 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 overcome 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 [29]. A series of seminal works further studied the properties of the sparse grid method and completed the framework [4, 9, 8, 37]. 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 slower in d. As shown by Zenger [37], the total number of points is $O\Big(N\big(\log(N)\big)^{d-1}\Big)$ for $N = 2^n$, where n is the discretization level. 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 accelerated version of ASK.

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 are utilized. In fact, the order of the error only increases slightly [37]. The polynomial interpolation in SASK borrows the ideas from [11] 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 $\xi_j = \cos\left(\frac{j\pi}{n-1}\right)$ for $j \in \{0, 1, \dots, n-1\}$. The construction of the sparse grids in a multi-dimensional domain builds on the uni-dimensional 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 in one dimension, thus further dictating the overall degree of approximation. In particular, $n = 2^\kappa + 1$ when $\kappa \geq 1$ and $n = 1$ 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:

$$\begin{aligned} &1)\ i_1 = 1, i_2 = 1, \quad 2)\ i_1 = 2, i_2 = 1, \quad 3)\ i_1 = 3, i_2 = 1, \\ &4)\ i_1 = 1, i_2 = 2, \quad 5)\ i_1 = 1, i_2 = 3, \quad 6)\ i_1 = 2, i_2 = 2. \end{aligned} \tag{10}$$

Let $\mathcal{S}(\cdot,\cdot)$ be the tensor product of two sets of points. Then, the combinations (i_1, i_2) in eq. (10) provide $\mathcal{S}(\mathcal{N}_{i_1}, \mathcal{N}_{i_2})$. For example, $\mathcal{S}(\mathcal{N}_1, \mathcal{N}_3) = \left\{(0,0), (0,-1), (0,1), (0, -\frac{\sqrt{2}}{2}), (0, \frac{\sqrt{2}}{2})\right\}$. 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 the appendix 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 [8, 11]. 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 $\mathcal{A}_i \cap \mathcal{A}_{j\neq i} = \varnothing$. 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_n are given by a recurrence relation: $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ 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 uni-dimensional 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 $\Psi_l(\boldsymbol{x})$ by 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 denote the unknown coefficients. Given the grid points $\{\boldsymbol{\xi}_l\}_{l=1}^N$, we can write

$$\begin{bmatrix} h^N(\boldsymbol{\xi}_1) \\ h^N(\boldsymbol{\xi}_2) \\ \vdots \\ h^N(\boldsymbol{\xi}_N) \end{bmatrix} = \begin{bmatrix} \Psi_1(\boldsymbol{\xi}_1) & \Psi_2(\boldsymbol{\xi}_1) & \dots & \Psi_N(\boldsymbol{\xi}_1) \\ \Psi_1(\boldsymbol{\xi}_2) & \Psi_2(\boldsymbol{\xi}_2) & \dots & \Psi_N(\boldsymbol{\xi}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_1(\boldsymbol{\xi}_N) & \Psi_2(\boldsymbol{\xi}_N) & \dots & \Psi_N(\boldsymbol{\xi}_N) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}. \tag{11}$$

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 eq. (7) as,

$$g(\boldsymbol{x}(t+t_0)) \approx g_N(\boldsymbol{x}(t+t_0)) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}(t_0)) e^{\tilde{\lambda}_j t}, \tag{12}$$

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 $\boldsymbol{x}_0 \coloneqq \boldsymbol{x}(t_0) = \boldsymbol{x}(0)$. The property of the infinitesimal generator eq. (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

$$\mathcal{K}\varphi = \mathbf{f} \cdot \nabla\varphi = f_1\frac{\partial\varphi}{\partial x_1} + f_2\frac{\partial\varphi}{\partial x_2} + \ldots + f_d\frac{\partial\varphi}{\partial x_d}. \tag{13}$$

The polynomial approximation φ_j^N in eq. (12) can be obtained based on eq. (13).

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 $\boldsymbol{\xi}_l \coloneqq (\xi_{l_1}, \xi_{l_2}, \ldots, \xi_{l_d}) \in \mathbb{R}^d$. Replacing h in eq. (11) 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, let 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 $(\partial_{x_i}\varphi^N)_l \coloneqq \frac{\partial\Psi}{\partial x_i}(\boldsymbol{\xi}_l)$. Let $\mathbf{K} \in \mathbb{R}^{N\times N}$ be the finite-dimensional approximation of $\mathcal{K}$. Given the dynamics $\mathbf{f} = [f_1, f_2, \ldots, f_d]^\intercal$, eq. (13) implies

$$\mathbf{K}\boldsymbol{\varphi}^N = \sum_{i}^{d} diag\big(f_i(\boldsymbol{\xi}_1), \ldots, f_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i\boldsymbol{w}. \tag{14}$$

3.3 Eigen-decomposition

With the discretized Koopman operator, we intend to obtain the eigenfunction values using the eigen-decomposition. 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 $\mathbf{K}\boldsymbol{\varphi}_j^N = \tilde{\lambda}_j\boldsymbol{\varphi}_j^N$. By eq. (14) and $\boldsymbol{\varphi}_j^N = \mathbf{M}\boldsymbol{w}_j$, we have

$$\sum_{i}^{d} diag\big(f_i(\boldsymbol{\xi}_1), \ldots, f_i(\boldsymbol{\xi}_N)\big)\, \mathbf{G}_i\boldsymbol{w}_j = \tilde{\lambda}_j\mathbf{M}\boldsymbol{w}_j. \tag{15}$$

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

$$\mathbf{U}\mathbf{W} = \mathbf{M}\mathbf{W}\boldsymbol{\Lambda}, \tag{16}$$

where $\mathbf{W} := \begin{bmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 & \ldots & \boldsymbol{w}_N \end{bmatrix}, \boldsymbol{\Lambda} := \operatorname{diag}\left(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_N\right)$. 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 eigen-decomposition. This is because matrix $\mathbf{M}$ is identity matrix $\mathbf{I}$ in ASK as it uses 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 eq. (16) is degenerated as $\mathbf{U}\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda}$. 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 [16]). 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 eigen-decomposition 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 multi-dimensional 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 $r := r_1 = r_2 = \ldots = r_d$ 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

$$g_N(\boldsymbol{x}(t)) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}_0) e^{\tilde{\lambda}_j t}. \tag{17}$$

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

$$\boldsymbol{x}(t) = \sum_{j=1}^{N} \tilde{c}_j \varphi_j^N(\boldsymbol{x}_0) e^{\tilde{\lambda}_j t} = \sum_{j=1}^{N} \tilde{c}_j (\boldsymbol{\varphi}_j^N)_1 e^{\tilde{\lambda}_j t}. \tag{18}$$

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 section 3.2– section 3.4.

Specifically, we set a series of check points in the time span $0 < \tau_1 < \tau_2 < \ldots < \tau_n < T$. On each of the point, 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

$$R_i := [L_i + \gamma r, U_i - \gamma r], \tag{19}$$

where L_i, U_i are the lower and upper bounds, r is the radius mentioned in section 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 $R_1 \times R_2 \times \ldots \times R_d$ 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 eigen-decomposition 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 eq. (18).

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 [16], 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, section 3.2– section 3.4 finds the finite-dimensional approximation of the Koopman operator, and approximates the eigenfunctions and eigenvalues. The solution is obtained by a linear combination of the eigenfunctions. In order to

Algorithm 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 $L_i = x_0^i - r, U_i = x_0^i + r$ and set neighborhood $R_i = [L_i + \gamma r, U_i - \gamma r]$ for $i = 1, 2, ..., d$.
 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 eigen-decomposition 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 in section 3.4.
 6: for $k = 1, 2, 3, \ldots, n$ do
 7:     Let $\nu_j$ be the first element of the $j$th column of $\mathbf{\Phi}$. Construct solution at time $\tau_k$ as $\boldsymbol{x}(t_\tau) = \sum_j \mathbf{C}(j,:)\nu_j e^{\tilde{\lambda}_j(\tau_k - \tau_{k-1})}$, where $\mathbf{C}(j,:)$ is the $j$th row of $\mathbf{C}$.
 8:     if $x_i(\tau_k) \notin R_i$ for any $i$ then
 9:         Set $L_i = x_i(\tau_k) - r$, $U_i = x_i(\tau_k) + r$ and $R_i = [L_i + \gamma r, U_i - \gamma r]$.
10:         Repeat steps 3 - 5.
11:     end if
12: end for
13: return $\boldsymbol{x}(T) = \sum_j \mathbf{C}(j,:)\nu_j e^{\tilde{\lambda}_j(T - \tau_n)}$.
```

preserve accuracy as time evolves, the adaptivity is added into the numerical scheme. The complete algorithm is summarized in algorithm 1. Of note, the construction of the sparse grid is based on the reference domain $[-1, 1]$ in practice. We then only need to re-scale the sparse grid points and matrices $\mathbf{G}_i$ on the reference domain by $\frac{U_i - L_i}{2}(\mathbf{\Xi}_i + 1) + L_i$ and $\frac{2\mathbf{G}_i}{U_i - L_i}$ respectively, so that they match the real domain $[L_i, U_i]$.

4 Numerical Results

The performance of SASK is demonstrated by multiple ODEs and PDEs in this section. In particular, the ODEs are two- and three-dimensional widely applied nonlinear dynamical systems. To solve a PDE with SASK, we exploit the semi-discrete 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. These tests investigate the accuracy and the efficiency of SASK compared with ASK as well as conventional ODE solver (here we use fourth order Runge-Kutta (RK4) method for comparison). Such study includes the impact of the approximation level κ, the number of check points n, and the radius r. For ODEs that do not have a closed-form solution, Verner's ninth order Runge-Kutta (RK9) method [33] is implemented to provide the reference solutions. Similarly, for the PDEs without a close-form solution, we apply a high-accuracy spectral collocation method to generate the reference solution.

4.1 ODEs

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

4.1.1 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 [2]:

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

The initial state is set $\boldsymbol{x}(0) = (10, 5)^\top$, and the terminal time is $T = 20$. The three tests corresponding to κ, n, r employ the following parameters:

(a) test of κ: $n = 200, r = 0.75$;

(b) test of n: $\kappa = 3, r = 0.2$;

(c) test of r: $\kappa = 3, n = 200$.

The parameter γ is set to 0.5 in the three tests. The results show that the error falls exponentially with the level of approximation. Differing from ASK, SASK is more sensitive to the number of check points. This is not surprising since the sparse grid is less accurate than the full grid and requires more careful examinations. It is worth emphasizing that the accuracy decreases if the radius is set too small or too large. Hence, we observed a v-shaped pattern in the test of radius.

4.1.2 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 g is the gravity acceleration, L is the length of the pendulum, and θ is the displacement. Setting $L = g$ for convenience, we can further convert the ODE into an equivalent two-dimensional first order ODE,

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

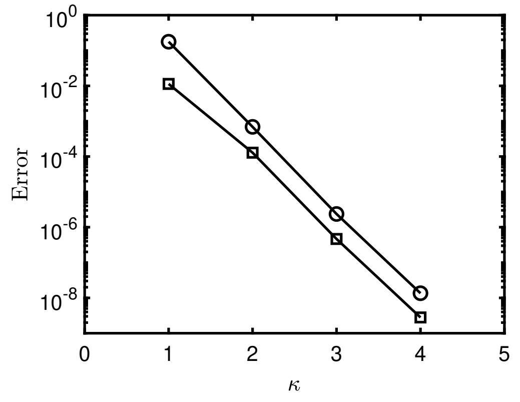
Here, $x_1 := \theta, x_2 := \frac{\mathrm{d}\theta}{\mathrm{d}t}$. The initial state is $\boldsymbol{x}(0) = \left(-\frac{\pi}{4}, \frac{\pi}{6}\right)^\top$, and the terminal time $T = 20$. For the simple pendulum, we fix $\gamma = 0.2$ in the tests with other parameters listed below:

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

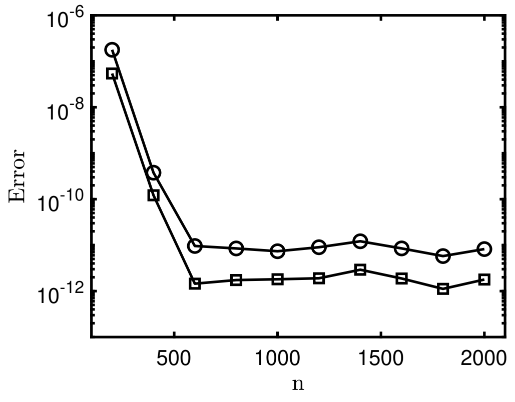
(b) test of n: $\kappa = 3, r = 0.1$;

(c) test of r: $\kappa = 3, n = 200$.

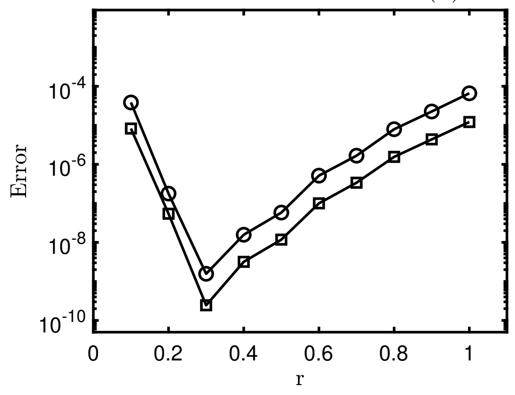
As shown in fig. 2, SASK achieves an exponential convergence with respect to κ in this example as well. However, the number of check points does not have a significant impact on the performance. Again, the radius cannot be set too small or too large.



(a) Approximation level

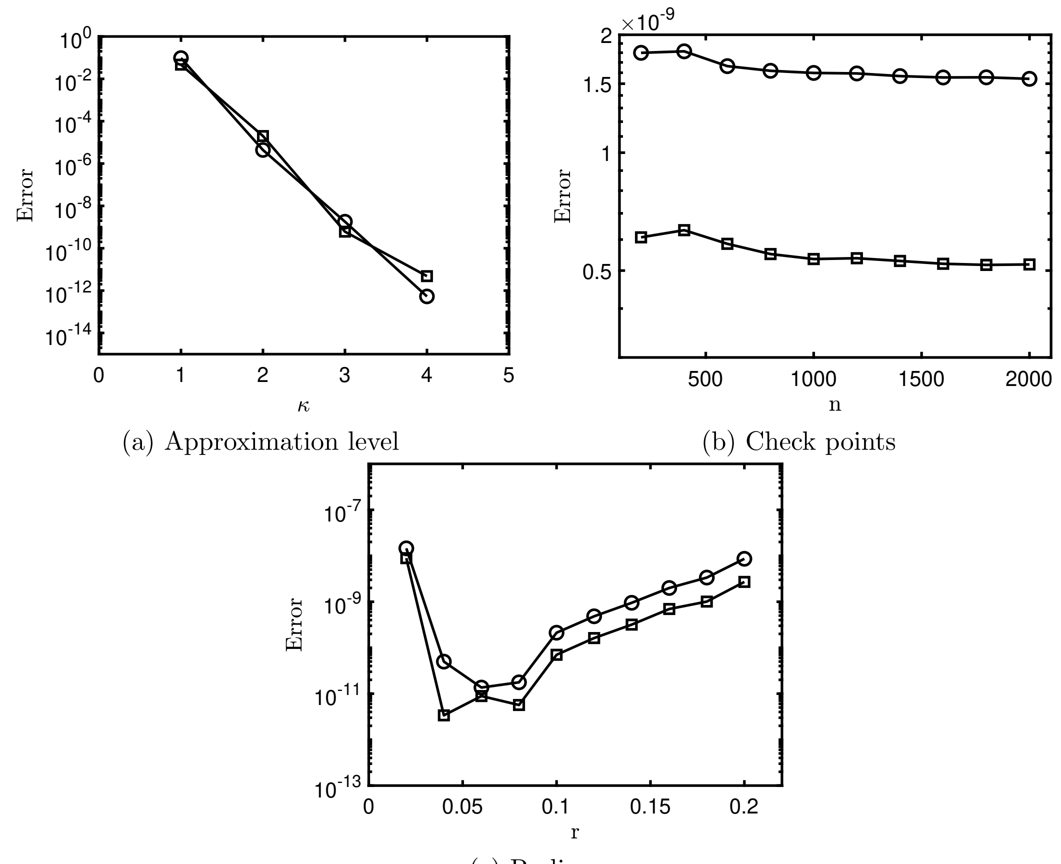


(b) Check points



(c) Radius

Figure 1: Lotka-Volterra Model: $\bigcirc, \square$ denote x_1, x_2, respectively



(a) Approximation level

(b) Check points

(c) Radius

Figure 2: Simple Pendulum: $\circ, \square$ denote x_1, x_2, respectively

4.1.3 Limit cycle

The limit cycle can be applied to describe the oscillatory patterns [34], 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: $x_1(t) = \cos(t + \arcsin(x_1(0)))$, $x_2(t) = \sin(t + \arcsin(x_2(0)))$. The parameters in the experiments are specified as follows:

(a) test of κ: $n = 200, r = \frac{\sqrt{2}}{20}$;

(b) test of n: $\kappa = 3, r = \frac{\sqrt{2}}{20}$;

(c) test of r: $\kappa = 3, n = 400$.

We set $\boldsymbol{x}(0) = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)^{\top}, T = 20, \gamma = 0.2$ in the above tests. Similar to the Lotka-Volterra case, the accuracy directly depends on the approximation level and the number of check points. Moreover, the radius needs to fall in an appropriate range for the convergence.

Besides the solution at T, we also computed the solutions at $M = 200$ equidistant time points on a time mesh over $[0, T]$, so as to compare SASK with RK4. Their performance is measured by the errors defined as $|x_{\text{SASK}} - x^*(t)|$ and $|x_{\text{RK4}} - x^*(t)|$, where x^* denotes the closed-form solutions. SASK employs $\kappa = 3, r = \frac{\sqrt{2}}{20}, \gamma = 0.2$. Of note, we overlap the check points with the time points of RK4 for comparison. The results are shown in fig. 4. Although SASK is not so accurate as ASK, it still outperformed RK4, achieving a 10^{-8} level of accuracy.

4.1.4 Kraichnan-Orszag model

The Kraichnan-Orszag problem was introduced in [23]. It induces a chaotic three-dimensional dynamical system, defined by the following equations,

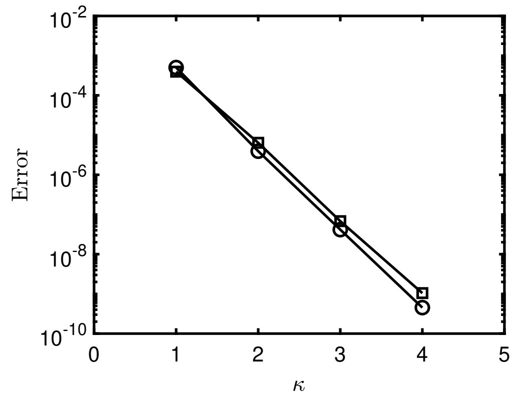
$$\begin{aligned}\frac{\mathrm{d}x_1}{\mathrm{d}t} &= x_2 x_3,\\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= x_1 x_3,\\ \frac{\mathrm{d}x_3}{\mathrm{d}t} &= -2x_1 x_2.\end{aligned}$$

For this model, $\boldsymbol{x}(0) = (1, 2, -3)^{\top}, T = 20, \gamma = 0.15$. The other parameters are specified as follows,

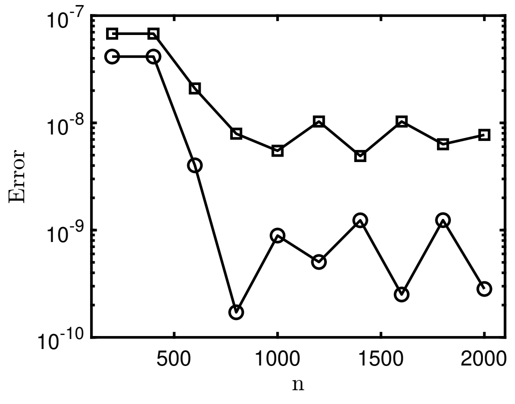
(a) test of κ: $n = 400, r = 0.2$;

(b) test of n: $\kappa = 3, r = \frac{\sqrt{2}}{20}$;

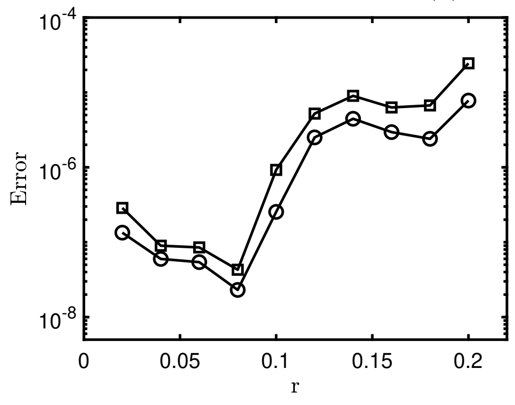
(c) test of r: $\kappa = 3, n = 200$.



(a) Approximation level

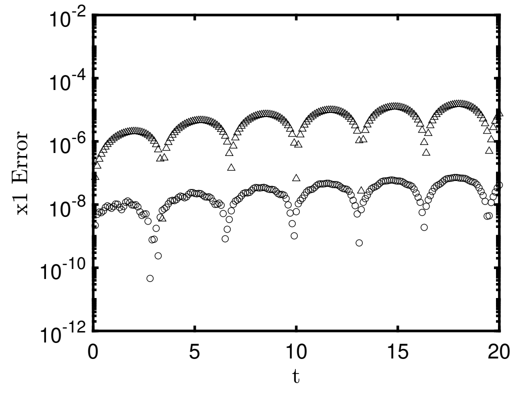


(b) Check points

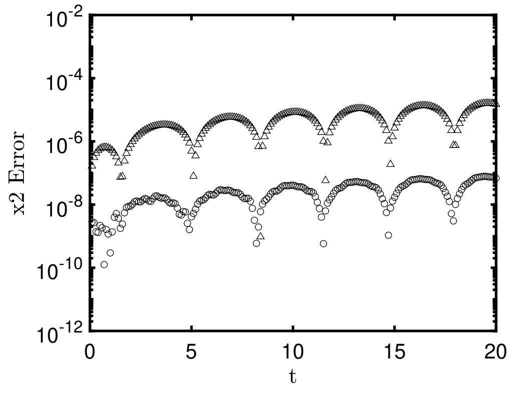


(c) Radius

Figure 3: Limit Cycle: $\bigcirc, \square$ denote x_1, x_2, respectively



(a) Error of x_1



(b) Error of x_2

Figure 4: Limit Cycle Full Solution Errors: $\bigcirc$ denotes SASK and $\triangle$ denotes RK4

The test of κ only includes $\kappa = 1, 2, 3$ because SASK encounters numerical issues when $\kappa = 4$, which leads to a drastic decease in accuracy. This is because the Kraichnan-Orszag dynamics are less smooth. As the states strongly oscillate as time evolves, one needs to frequently examine the validity of the neighborhood of the intermediate states. Therefore, more check points are utilized than the aforementioned models. Moreover, the "optimal" radius is slightly larger, illustrated by fig. 5, because of the chaotic evolution.

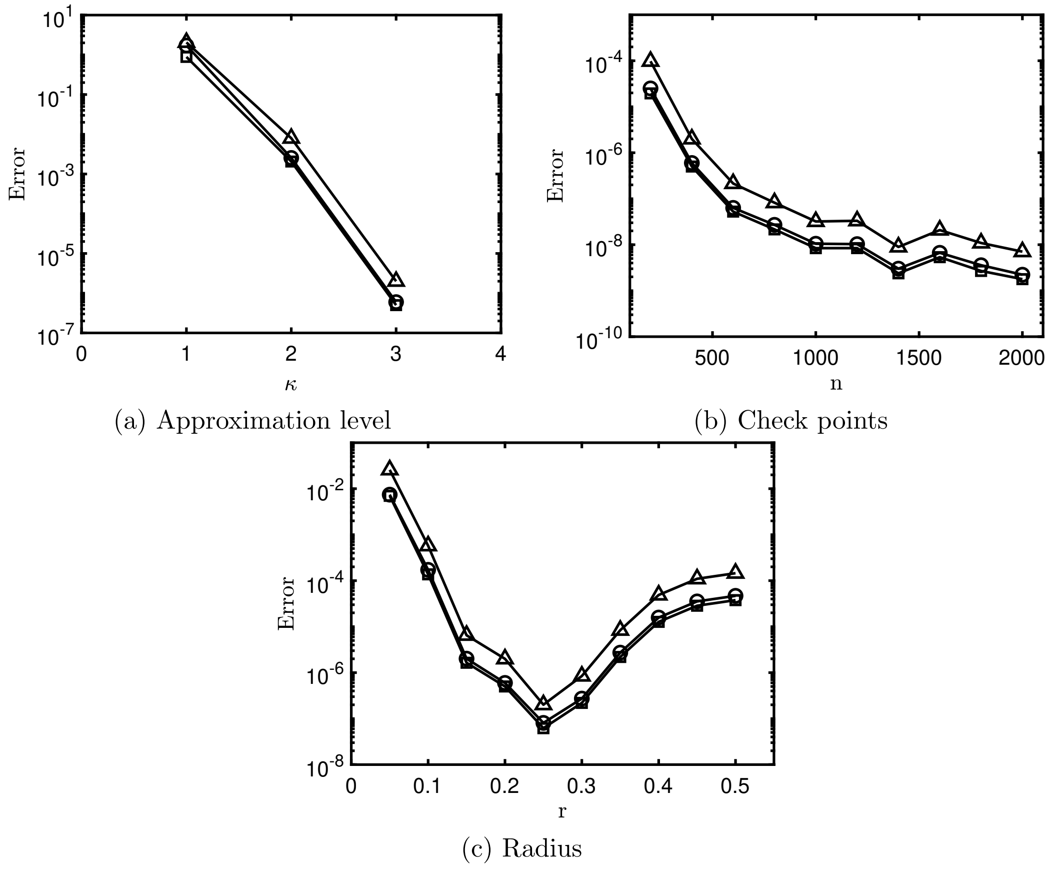


Figure 5: Kraichnan-Orszag Model:○, □, △ denote x_1, x_2, x_3, respectively

4.1.5 Lorenz attractor

The Lorenz attractor [17] was introduced to model the turbulence in dynamic flows. The system is highly chaotic and exhibits behaviors that pose challenges to solve numerically. The governing equations are shown below:

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

The terminal time is $T = 10$, and the initial state is $\boldsymbol{x}(0) = (5,5,5)^{\intercal}$. We show the parameters used as follows,

(a) test of κ: $n = 1000, r = 1$;

(b) test of n: $\kappa = 3, r = 1$;

(c) test of r: $\kappa = 3, n = 1000$.

The parameter γ is fixed to 0.5 in all three tests. Due to the chaotic behavior, a large number of check points is required to guarantee high accuracy. Also, a relatively large radius leads to better results, as demonstrated in fig. 6.

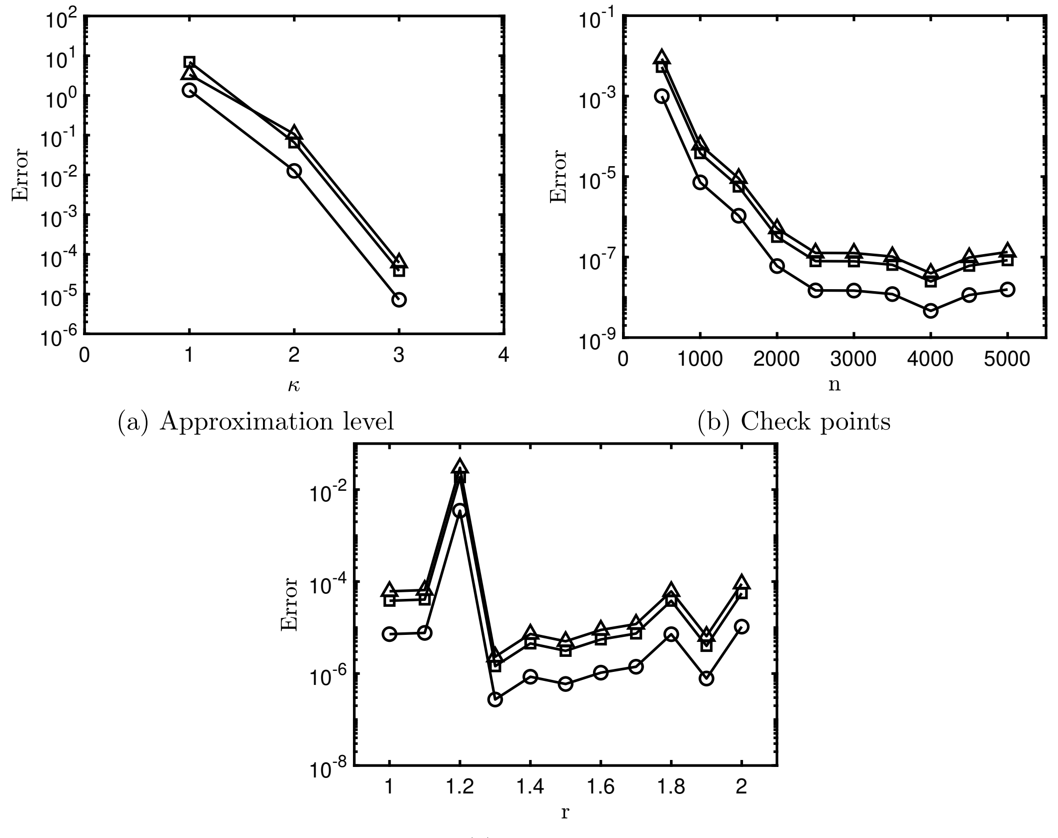
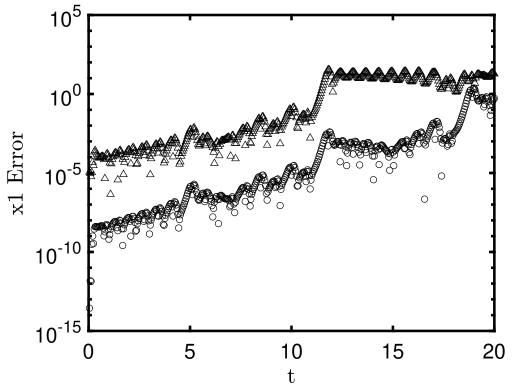
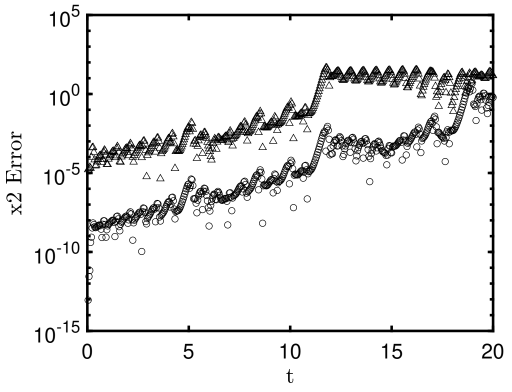


Figure 6: Lorenz Attractor: $\circ, \square, \triangle$ denote x_1, x_2, x_3, respectively

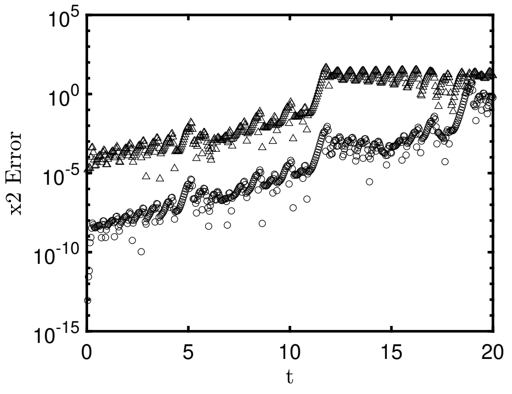
For the Lorenz attractor, we also compared SASK with RK4 over the time span $[0, 20]$. Specifically, RK4 leverages $M = 2000$ time steps, while SASK has the parameters $n = 2000, \kappa = 3, r = 1, \gamma = 0.75$. The check points overlap with the time points of the time mesh so that we can compare the solutions of SASK and RK4 at these points. Since the Lorenz attractor does not have analytical solutions, RK9 is applied to serve as reference. Notably, we set $M = 20000$ time points for RK9 to guarantee the accuracy of its solutions. Figure 7 shows that SASK constantly outperformed RK 4 over the time span, although it is not so accurate near $T = 20$ with the error exceeding 10^{-1}.



(a) Error of x_1



(b) Error of x_2



(c) Error of x_3

Figure 7: Lorenz Attractor Full Solution Errors: $\circ$ denotes SASK and $\triangle$ denotes RK4

4.2 PDEs

SASK can also be employed to solve PDEs in the semi-discrete form, induced by discretization schemes in space. To illustrate its effectiveness, we implemented numerical experiments on four PDEs, the details of which will be exhibited subsequently. Specifically, the spacial discretization is based on Fourier collocation method (see e.g., [28, 10, 31]) as we impose *periodic* boundary conditions to all the PDEs. The MATLAB code generating differentiation matrices can be found in [31]. 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 $2m + 1$. Nevertheless, the results turn out to be accurate with such a low approximation level. Suppose the degree of freedom in space is m, then the ODE system is m-dimensional. We denote $y \in \mathbb{R}^m$ the SASK solution at different spatial grid points, and 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 $\|y - y^*\|_\infty$.

4.2.1 Advection equation

We consider the following advection equation with an initial condition:

$$\begin{aligned} &\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \quad x \in [0,1], \\ &u(x,0) = 0.2 + \sin(\cos(4\pi x)). \end{aligned} \tag{20}$$

The close-form solution is $u(x,t) = u(x-t,0)$. The collocation points in space are set as $x_j = \frac{j}{32}, j = 0,1,\dots,32$, which leads to a 32-dimensional ODE system. SASK applied the following set of parameters: $n = 10, r = 1, \gamma = 0.2$. The errors at $T = 10$ are $e_{L_2} = 1.89 \times 10^{-12}$ and $e_{L_\infty} = 2.89 \times 10^{-12}$. Figure 8 illustrates the SASK solutions compared with the exact solutions. We also observed that the accuracy did not decrease significantly if $n = 1$. This is because the dynamics are linear after semi-discretization, and adaptivity is barely triggered.

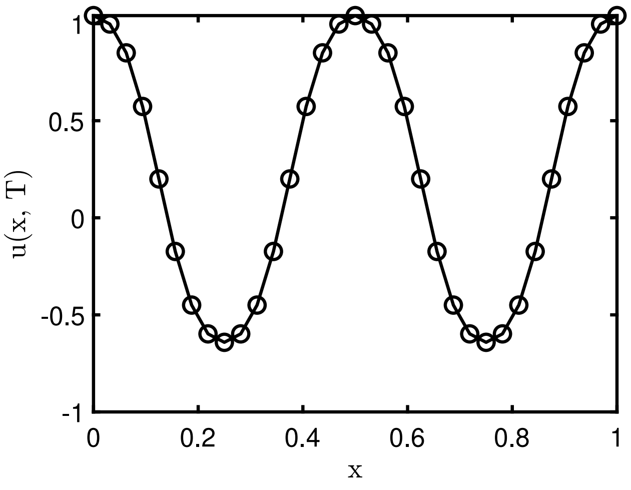


Figure 8: Advection equation: $\circ, -$ denote SASK and reference, respectively

4.2.2 Heat equation

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

$$\begin{aligned} &\frac{\partial u}{\partial t} = \frac{1}{80\pi^2}\frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1], \\ &u(x,0) = \sin(4\pi x). \end{aligned} \tag{21}$$

This equation has a closed-form solution $u(x,t) = \sin(4\pi x)e^{-0.2t}$. Specifying parameters $m = 32, n = 10, r = 0.1, \gamma = 0.2$, we obtain SASK numerical solutions at $T = 10$. The comparison between the exact solutions with numerical solutions is exhibited in fig. 9. In particular, $e_{L_2} = 8.69 \times 10^{-16}$ and $e_{L_\infty} = 9.08 \times 10^{-15}$. As in the advection equation case, the semi-discrete form is a linear ODE system, and we observed that SASK achieved high accuracy even when $n = 1$.

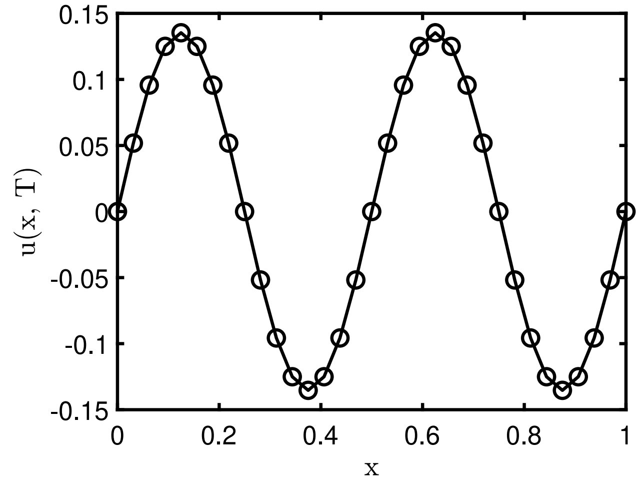


Figure 9: Heat equation: ○, – denote SASK and reference, respectively

4.2.3 Burgers equation

Next, a viscous Burgers equation with an initial condition is considered:

$$\begin{aligned} &\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1], \\ &u(x,0) = 0.2 + \sin(2\pi x). \end{aligned} \tag{22}$$

The advection term uu_x is treated in the conservation form, i.e., $\frac{1}{2}(u^2)_x$. 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 $n = 50, r = 0.1, \gamma = 1$. With $\nu = 0.005$ and $T = 1$, SASK yields $e_{L_2} = 8.85 \times 10^{-4}$, $e_{L_\infty} = 1.80 \times 10^{-3}$. The result is presented in fig. 10, which indicates that before the discontinuity is fully developed, the solution is accurate. Of note, if we keep increasing T, we will observe the Gibbs phenomenon near the discontinuity.

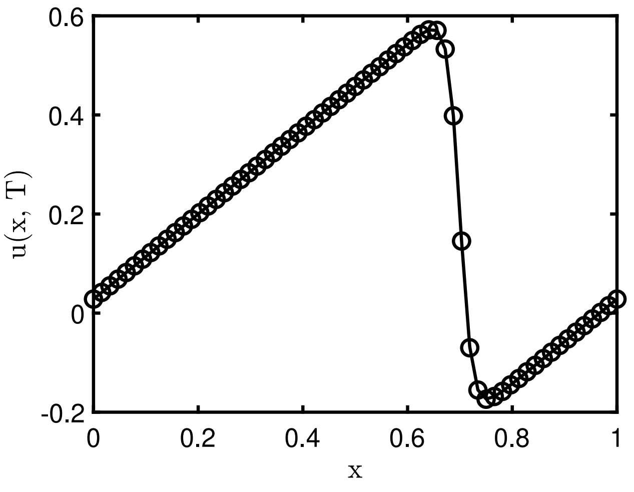


Figure 10: Burgers Equation: $\circ, -$ denote SASK and reference, respectively

4.2.4 Korteweg-de Vries equation

The last example is the Korteweg-de Vries (KdV) equation with a solitary wave solution:

$$\begin{aligned} &\frac{\partial u}{\partial t} + \beta u \frac{\partial u}{\partial x} + \mu \frac{\partial^3 u}{\partial x^3} = 0, \quad x \in \mathbb{R}, \\ &u(x, 0) = \frac{3c}{\beta} sech^2 \left(\frac{1}{2} \sqrt{c/\mu} x \right). \end{aligned} \tag{23}$$

The closed-form solution is $u(x,t) = \frac{3c}{\beta} sech^2 \left(\frac{1}{2} \sqrt{c/\mu}(x - ct) \right)$. Here, β, μ, c are constants, and c is the wave speed. In this test, we set $c = 0.5, \beta = 3, \mu = 9$. In general, the solution decays to zeros for $|x| >> 1$. Therefore, numerically we solve this equation in a finite domain $[-p, p]$ with periodic boundary condition. In this example we set $p = 30$. Further, as shown in [7, 28], 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 eq. (23) 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 $m = 100$ since the behavior of the KdV equation requires finer spatial grids. The parameter related to SASK are $n = 100, r = 0.1, \gamma = 0.8$. SASK computed the solutions at $T = 10$, giving the illustration in fig. 11. The errors are $e_{L_2} = 1.60 \times 10^{-4}$ and $e_{L_\infty} = 1.40 \times 10^{-4}$.

4.3 Efficiency comparison

4.3.1 Comparison on ODEs

For the ODEs above, we compared the performance of ASK and SASK using the same set of parameters. The results are summarized in Table 1. Running time is measured in seconds. As expected, SASK could not reach the high accuracy as ASK did, especially for chaotic systems, i.e. the Kraichnan-Orszag model and the Lorenz attractor. However, it ran much faster, as ASK used from 3× to 10× the time of SASK for the three two-dimensional systems. In the three-dimensional system, the acceleration is around 100×.

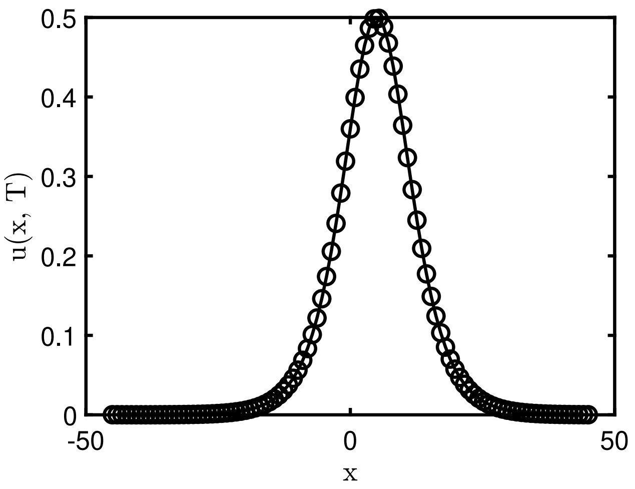


Figure 11: KdV Equation: ○, – denote SASK and reference, respectively

Table 1: SASK vs ASK

Models	SASK_e1	SASK_e2	SASK_e3	SASK_t	ASK_e1	ASK_e2	ASK_e3	ASK_t
LV	5.81e-08	1.18e-08	NaN	0.10	3.76e-10	9.37e-11	NaN	0.35
SP	2.80e-07	9.05e-08	NaN	0.03	9.00e-11	3.41e-11	NaN	0.14
LC	1.20e-09	1.03e-08	NaN	0.08	1.75e-13	4.30e-13	NaN	0.70
KO	1.01e-04	8.19e-05	3.16e-04	0.50	5.24e-07	4.16e-07	1.68e-06	43.73
LO	1.02e-03	5.33e-03	8.59e-03	0.69	9.78e-08	5.21e-07	8.36e-07	72.21

4.3.2 Comparison on PDEs

To demonstrate the efficiency of SASK compared with RK4, we target the wall time of the two methods based on the PDEs. Particularly, RK4 leverages the Fourier differentiation matrix to the solve the semi-discrete form after spatial discretization. The parameters used in the four PDEs are specified as follows:

(a) Adevection equation: $m = 32, T = 100, n = 1, r = 1, \gamma = 0.2$;

(b) Heat equation: $m = 32, T = 10, n = 10, r = 0.1, \gamma = 0.5$;

(c) Burgers equation: $\nu = 0.005, m = 64, T = 1, n = 100, r = 0.1, \gamma = 1$;

(d) KdV equation: $c = 0.5, \beta = 3, \mu = 9, p = 45, m = 100, T = 10, n = 100, r = 0.1, \gamma = 0.8$.

We summarize the results in table 2. The first row of each PDE is the wall 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 semi-discrete form are linear so SASK needs only a small number of check points and updates to be accurate. This reduces the computational time of SASK, which makes SASK much faster than RK4. In contrast, Burgers equation and KdV equation require more updates so SASK takes longer to preserve accuracy. As in the Burgers equation result, SASK spends around 10 times the time of RK4 to reach a comparable level of accuracy. Hence, SASK gains an edge over RK4 when T is large, or when the dynamics are not so nonlinear that it allows for a small number of updates.

Table 2: SASK vs RK4

		SASK	RK4
Advection Equation	t	0.0581	1.2473
	L_2	2.80e-11	1.49e-07
	L_∞	3.02e-11	1.78e-07
Heat Equation	t	0.0855	1.1353
	L_2	7.57e-15	9.24e-15
	L_∞	1.19e-15	1.85e-15
Burgers Equation	t	1.8669	0.1830
	L_2	4.94e-04	4.88e-04
	L_∞	6.13e-04	4.96e-04
KdV Equation	t	0.8901	3.5348
	L_2	1.5972e-04	1.5968e-04
	L_∞	1.4030e-04	1.4090e-04

5 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 of solving PDEs efficiently by solving semi-discrete systems. In particular, our numerical results demonstrate that if the semi-discrete 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 because SASK is a high order method for solving dynamical systems. Further, by select the adaptivity criteria carefully, SASK can solve highly nonlinear PDEs like the KdV equation and deal with with large total variation in the solution like the Burgers' equation. Moreover, we only used level-1 sparse grid method in SASK to obtain good results in the illustrative examples. In our future work, we will further investigate the selection of adaptivity paramters and the requirement on the accuracy level of the sparse grid method.

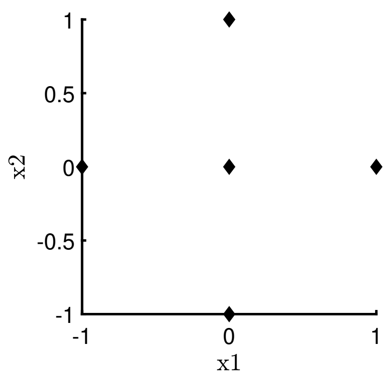
A Sparse grid illustration

Figure 12 is an illustration of sparse grids and their full grid counterparts. To give a brief view of the growth in the grid size, table 3 shows that the full grid grows much faster than the sparse grid. Notably, the sparse grid grows surprisingly slow in practice, although its theoretical order is exponential in d.

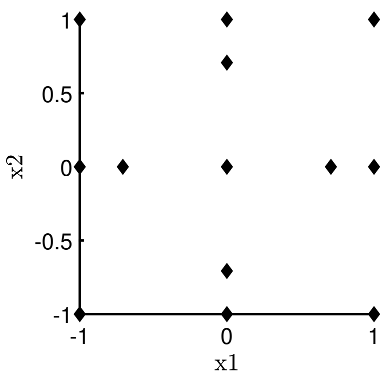
References

[1] Travis Askham and J Nathan Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 17(1):380–416, 2018.

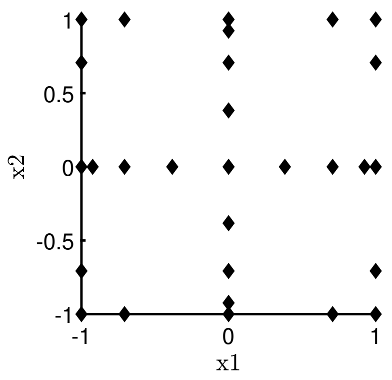
[2] Immanuel M Bomze. Lotka-volterra equation and replicator dynamics: a two-dimensional classification. *Biological cybernetics*, 48(3):201–211, 1983.



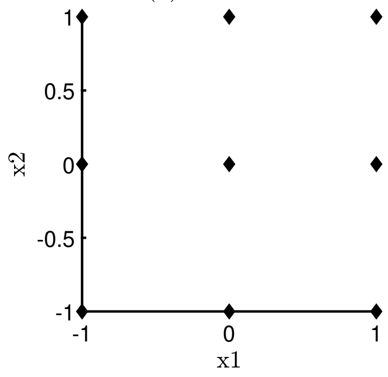
(a) $\kappa = 1$



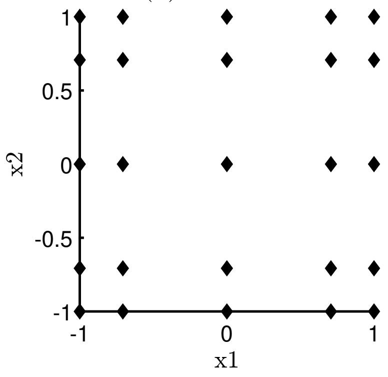
(b) $\kappa = 2$



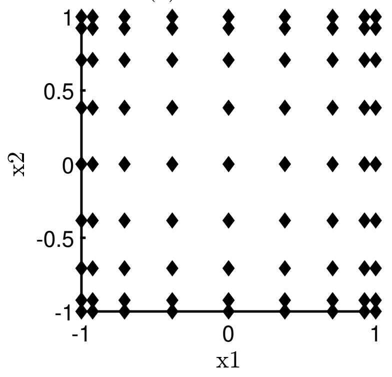
(c) $\kappa = 3$



(d) $N = 3$



(e) $N = 5$



(f) $N = 9$

Figure 12: Sparse Grid (First Row) - Full Grid (Second Row)

Table 3: Grid Growth

d	$\kappa = 1$	$N = 3$	$\kappa = 2$	$N = 5$	$\kappa = 3$	$N = 9$
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

[3] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern Koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[4] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[5] Akshunna S Dogra and William Redman. Optimizing neural networks via Koopman operator theory. *Advances in Neural Information Processing Systems*, 33:2087–2097, 2020.

[6] Bengt Fornberg. *A practical guide to pseudospectral methods*. Number 1. Cambridge university press, 1998.

[7] Bengt Fornberg and Gerald Beresford Whitham. A numerical and theoretical study of certain nonlinear wave phenomena. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 289(1361):373–404, 1978.

[8] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numerical algorithms*, 18(3):209–232, 1998.

[9] Michael Griebel, Michael Schneider, and Christoph Zenger. A combination technique for the solution of sparse grid problems. 1990.

[10] Jan S Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.

[11] Kenneth L Judd, Lilia Maliar, Serguei Maliar, and Rafael Valero. Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. *Journal of Economic Dynamics and Control*, 44:92–123, 2014.

[12] Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315, 1931.

[13] Milan Korda, Mihai Putinar, and Igor Mezić. Data-driven spectral analysis of the Koopman operator. *Applied and Computational Harmonic Analysis*, 48(2):599–629, 2020.

[14] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[15] J Nathan Kutz, Xing Fu, and Steven L Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.

[16] Bian Li, Yi-An Ma, J Nathan Kutz, and Xiu Yang. The adaptive spectral koopman method for dynamical systems. *arXiv preprint arXiv:2202.09501*, 2022.

[17] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[18] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

[19] Igor Mezić. Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry. *Journal of Nonlinear Science*, 30(5):2091–2145, 2020.

[20] Hiroya Nakao and Igor Mezić. Spectral analysis of the Koopman operator for partial differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(11):113131, 2020.

[21] J Nathan Kutz, Joshua L Proctor, and Steven L Brunton. Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems. *Complexity*, 2018, 2018.

[22] Marcos Netto and Lamine Mili. A robust data-driven Koopman kalman filter for power systems dynamic state estimation. *IEEE Transactions on Power Systems*, 33(6):7228–7237, 2018.

[23] Steven A Orszag and LR Bissonnette. Dynamical properties of truncated Wiener-Hermite expansions. *The Physics of Fluids*, 10(12):2603–2613, 1967.

[24] Jacob Page and Rich R Kerswell. Koopman analysis of Burgers equation. *Physical Review Fluids*, 3(7):071901, 2018.

[25] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[26] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[27] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[28] Jie Shen and Tao Tang. *Spectral and high-order methods with applications.* Science Press Beijing, 2006.

[29] Sergei Abramovich Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Doklady Akademii Nauk*, volume 148, pages 1042–1045. Russian Academy of Sciences, 1963.

[30] Amit Surana and Andrzej Banaszuk. Linear observer synthesis for nonlinear systems using Koopman operator framework. *IFAC-PapersOnLine*, 49(18):716–723, 2016.

[31] Lloyd N Trefethen. *Spectral methods in MATLAB.* SIAM, 2000.

[32] Jonathan H Tu, Clarence W Rowley, Dirk M Luchtenburg, Steven L Brunton, and J Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[33] James Hamilton Verner. Explicit Runge–Kutta methods with estimates of the local truncation error. *SIAM Journal on Numerical Analysis*, 15(4):772–790, 1978.

[34] Mathukumalli Vidyasagar. *Nonlinear systems analysis.* SIAM, 2002.

[35] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[36] Dan Wilson and Jeff Moehlis. Isostable reduction with applications to time-dependent partial differential equations. *Physical Review E*, 94(1):012211, 2016.

[37] Christoph Zenger and W Hackbusch. Sparse grids. In *Proceedings of the Research Workshop of the Israel Science Foundation on Multiscale Phenomenon, Modelling and Computation*, page 86, 1991.