

**ISE**

Industrial and  
Systems Engineering

# The Adaptive Spectral Koopman Method for Dynamical Systems

BIAN LI<sup>1</sup>, YI-AN MA<sup>2</sup>, J. NATHAN KUTZ<sup>3</sup>, AND XIU YANG<sup>1</sup>

<sup>1</sup>Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

<sup>2</sup>Halicioğlu Data Science Institute and Department of Computer Science and Engineering,  
University of California San Diego, San Diego, CA, USA

<sup>3</sup>Department of Applied Mathematics, University of Washington, Seattle, WA, USA

ISE Technical Report 23T-005



# The Adaptive Spectral Koopman Method for Dynamical Systems\*

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

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

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

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

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

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

---

\*Submitted to the editors on March 30, 2022.

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

<sup>†</sup>Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA ([bil215@lehigh.edu](mailto:bil215@lehigh.edu), [xiy518@lehigh.edu](mailto:xiy518@lehigh.edu))

<sup>‡</sup>Halicioğlu Data Science Institute & Department of Computer Science and Engineering, University of California San Diego, San Diego, CA ([yianma@ucsd.edu](mailto:yianma@ucsd.edu))

<sup>§</sup>Department of Applied Mathematics, University of Washington, Seattle, WA ([kutz@uw.edu](mailto:kutz@uw.edu))

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

44 On the computational side, most existing numerical schemes motivated by the Koopman  
45 operator are categorized as data-driven methods, as they use spatiotemporal data to approxi-  
46 mate a few of the leading Koopman eigenvalues, eigenfunctions, and modes. In particular, the  
47 emerging computational method *dynamics mode decomposition* (DMD) [24, 25, 30, 23, 14, 20,  
48 1] as well as its variant such as extended DMD (EDMD) [34] uses snapshots of a dynamical  
49 system to extract temporal features as well as correlated spatial activity via matrix decom-  
50 position techniques. DMD and EDMD produce results for any appropriately formatted set of  
51 data, but connecting these outputs to the Koopman operator requires additional knowledge  
52 about the nature of the underlying system in that the system should be autonomous. Later, a  
53 modified EDMD [33] was proposed to compensate for the effects of system actuation when it  
54 is used to explore state space during the data collection, reestablishing the connection between  
55 EDMD and the Koopman operator in this more general class of data sets. A review of many  
56 of the DMD variants for approximating the Koopman operator can be found in Brunton et  
57 al [4].

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

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

75 **2. Background.**

76 **2.1. Koopman operator.** Borrowing notions from [13], we consider an autonomous system  
77 described by the ordinary differential equations

78 (2.1) 
$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}),$$

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

86 (2.2) 
$$\mathbf{x}(t_0 + t) = \mathbf{F}_t(\mathbf{x}(t_0)) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(s)) ds.$$

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

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

90 
$$\mathcal{K}_t g(\mathbf{x}_0) = g(\mathbf{F}_t(\mathbf{x}_0)),$$

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

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

95 **2.2. Infinitesimal generator.** The Koopman spectral theory [17, 24] reveals properties  
96 that enable the Koopman operator to convert nonlinear finite-dimensional dynamics into linear  
97 infinite-dimensional dynamics. A key component in such spectral analysis is the infinitesimal  
98 generator (or generator for brevity) of the Koopman operator. Specifically, the generator of  
99 the Koopman operator  $\mathcal{K}_t$ , denoted as  $\mathcal{K}$ , is given by

100 (2.3) 
$$\mathcal{K}g = \lim_{t \rightarrow 0} \frac{\mathcal{K}_t g - g}{t}.$$

102 For any smooth function  $g$ , (2.3) implies that

103 (2.4) 
$$\mathcal{K}g(\mathbf{x}) = \frac{dg(\mathbf{x})}{dt} = \nabla g(\mathbf{x}) \cdot \frac{d\mathbf{x}}{dt}.$$

105 Denoting  $\lambda$  an eigenfunction of  $\mathcal{K}$  and  $\omega$  the eigenvalue associated with  $\lambda$ , we have

106 (2.5) 
$$\mathcal{K} \lambda(\mathbf{x}) = \omega \lambda(\mathbf{x}).$$

108 Thus,

109 (2.6) 
$$\omega \lambda(\mathbf{x}) = \mathcal{K} \lambda(\mathbf{x}) = \frac{d}{dt} \lambda(\mathbf{x}).$$

111 This implies that  $\mathbf{x}(t_0 + t) = e^{t\mathcal{K}}(\mathbf{x}(t_0))$ , i.e.,

$$112 \quad (2.7) \quad \mathcal{K}_t(\mathbf{x}(t_0)) = e^{t\mathcal{K}}(\mathbf{x}(t_0)).$$

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

115 Now suppose  $g$  exists in the function space spanned by all the eigenfunctions  $\phi_j$  (associated  
116 with eigenvalues  $\lambda_j$ ) of  $\mathcal{K}$ , i.e.,  $g(\mathbf{x}) = \sum_j c_j \phi_j(\mathbf{x})$ , then

$$117 \quad (2.8) \quad \mathcal{K}_t[g(\mathbf{x}(t_0))] = \mathcal{K}_t\left[\sum_j c_j \phi_j(\mathbf{x}(t_0))\right] = \sum_j c_j \mathcal{K}_t[\phi_j(\mathbf{x}(t_0))].$$

118 Hence,

$$120 \quad (2.9) \quad g(\mathbf{x}(t_0 + t)) = \sum_j c_j \phi_j(\mathbf{x}(t_0)) e^{j t}.$$

121 Similarly, if we choose a vector-valued observable  $\mathbf{g}: \mathcal{M} \rightarrow \mathbb{R}^d$  with  $\mathbf{g} := (g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_d(\mathbf{x}))^\top$ ,  
122 the system of observables becomes

$$123 \quad (2.10) \quad \frac{d\mathbf{g}(\mathbf{x})}{dt} = \mathcal{K}\mathbf{g}(\mathbf{x}) = \begin{bmatrix} \mathcal{K}g_1(\mathbf{x}) \\ \mathcal{K}g_2(\mathbf{x}) \\ \vdots \\ \mathcal{K}g_d(\mathbf{x}) \end{bmatrix} = \sum_j \lambda_j \phi_j(\mathbf{x}) \mathbf{c}_j,$$

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

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

$$134 \quad x(t) = \sum_{j=1}^{\infty} c_j \phi_j(x_0) e^{j t} = \sum_{j=1}^{\infty} c_j x_0^j e^{j \mu t}.$$

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

137 **3. Adaptive Spectral Koopman Method.** In this section, we introduce the adaptive spec-  
138 tral Koopman (ASK) method, which is a numerical method based on the Koopman operator  
139 and the spectral method to solve ODE systems. Before describing details of this method,  
140 we introduce the notations used in this algorithm. Let  $\mathbf{x}(t)$  denote the solution of an ODE  
141 system with an initial condition  $\mathbf{x}(t_0) = \mathbf{x}_0$ . Assume  $t_0 = 0$  in (2.2), we consider solutions  
142 in time interval  $[0, T]$  with  $T > 0$ . Letter  $n$  denotes the number of ‘‘check points’’ (see de-  
143 tails in [subsection 3.4](#)). The radius of the neighborhood of  $\mathbf{x}(t)$  is denoted by  $r$  while  $\delta$  is a  
144 parameter that controls the update of the neighborhood.

145 **3.1. Finite-dimensional approximation.** Based on the preliminaries introduced in sub-  
 146 section 2.2, we aim to identify the following truncated approximation of (2.9)

$$147 \quad (3.1) \quad g(\mathbf{x}(t)) \approx g_N(\mathbf{x}(t)) = \sum_{j=0}^N \tilde{c}_j \tilde{g}_j^N(\mathbf{x}_0) e^{\tilde{\lambda}_j t}.$$

148 where  $\tilde{g}_j^N$  are polynomial approximations of  $g_j$ ,  $\tilde{\lambda}_j$  and  $\tilde{c}_j$  approximate  $\lambda_j$  and  $c_j$ , respectively.  
 149 Next, because  $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$ , (2.4) and (2.6) indicate that for any eigenfunction  $\tilde{g}_j^N$ ,

$$150 \quad \mathcal{K} \tilde{g}_j^N = \mathbf{f} \cdot \nabla \tilde{g}_j^N = \left( f_1 \frac{\partial}{\partial x_1} + f_2 \frac{\partial}{\partial x_2} + \dots + f_d \frac{\partial}{\partial x_d} \right) \tilde{g}_j^N$$

$$151 \quad = \left( f_1 \frac{\partial}{\partial x_1} + f_2 \frac{\partial}{\partial x_2} + \dots + f_d \frac{\partial}{\partial x_d} \right) \tilde{g}_j^N.$$

152 Thus,

$$153 \quad (3.2) \quad \mathcal{K} = f_1 \frac{\partial}{\partial x_1} + f_2 \frac{\partial}{\partial x_2} + \dots + f_d \frac{\partial}{\partial x_d}.$$

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

158 (1) When  $d=1$ . Let  $\{x_i\}_{i=0}^N$  be the Gauss-Lobatto points and the polynomial interpolation  
 159 of  $\tilde{g}_j^N(\mathbf{x})$  is

$$160 \quad \tilde{g}_j^N(\mathbf{x}) \approx \tilde{g}_j^N(x) := \sum_{i=0}^N \tilde{g}_j^N(x_i) P_i(x),$$

161 where the basis functions  $P_j$  are Lagrange polynomials satisfying  $P_j(x_i) = \delta_{ij}$  and  $\delta_{ij}$   
 162 is the Kronecker delta function. Namely,  $\tilde{g}_j^N(x)$  is the projection of  $\tilde{g}_j^N(\mathbf{x})$  on the space  
 163 span $\{P_j(x)\}_{j=0}^N$ . Let  $\mathbf{N} = [\tilde{g}_j^N(x_0), \tilde{g}_j^N(x_1), \dots, \tilde{g}_j^N(x_N)]^T$ , we have

$$164 \quad (3.3) \quad \mathcal{K} \mathbf{N} = \text{diag}(\mathbf{f}(x_0), \mathbf{f}(x_1), \dots, \mathbf{f}(x_N)) \mathbf{D} \mathbf{N} := \mathbf{K} \mathbf{N},$$

165 where  $\mathbf{D}$  is the differentiation matrix associated with  $\{x_i\}_{i=0}^N$  and  $\mathbf{K}$  is an  $(N+1) \times (N+1)$   
 166 matrix. Here, we abuse the notation to let  $\mathcal{K} \mathbf{N} = [\mathcal{K} \tilde{g}_j^N(x_0), \mathcal{K} \tilde{g}_j^N(x_1), \dots, \mathcal{K} \tilde{g}_j^N(x_N)]^T$ ,  
 167 and similar notations are used in the following  $d=2, 3$  cases.

168 (2) When  $d=2$ . Let  $\{x_i\}_{i=0}^N$  and  $\{y_j\}_{j=0}^N$  be the Gauss-Lobatto points of  $x_1$  and  $x_2$ ,  
 169 respectively. Every eigenfunction  $\tilde{g}_j^N$  is now a bivariate function, whose polynomial  
 170 interpolation  $\tilde{g}_j^N$  is

$$171 \quad \tilde{g}_j^N(x_1, x_2) \approx \tilde{g}_j^N(x_1, x_2) := \sum_{i=0}^N \sum_{j=0}^N \tilde{g}_j^N(x_i, y_j) P_i(x_1) P_j(x_2).$$

172 Hence, we define a matrix  $\Phi^N$  as

$$173 \quad \Phi^N = \begin{bmatrix} \tilde{g}_j^N(x_0, y_0) & \tilde{g}_j^N(x_0, y_1) & \dots & \tilde{g}_j^N(x_0, y_N) \\ \tilde{g}_j^N(x_1, y_0) & \tilde{g}_j^N(x_1, y_1) & \dots & \tilde{g}_j^N(x_1, y_N) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{g}_j^N(x_N, y_0) & \tilde{g}_j^N(x_N, y_1) & \dots & \tilde{g}_j^N(x_N, y_N) \end{bmatrix}.$$

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

$$181 \quad \mathcal{K}\Phi^N = \mathbf{F}_1 \odot (\mathbf{D}_1 \Phi^N) + \mathbf{F}_2 \odot (\Phi^N \mathbf{D}_2^\top),$$

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

$$185 \quad \begin{aligned} \mathcal{K} \text{vec}(\Phi^N) &= \text{vec}(\mathbf{F}_1) \odot \left( (\mathbf{I} \otimes \mathbf{D}_1) \text{vec}(\Phi^N) \right) + \text{vec}(\mathbf{F}_2) \odot \left( (\mathbf{D}_2 \otimes \mathbf{I}) \text{vec}(\Phi^N) \right) \\ 186 \quad &= \left[ \text{diag}(\text{vec}(\mathbf{F}_1))(\mathbf{I} \otimes \mathbf{D}_1) + \text{diag}(\text{vec}(\mathbf{F}_2))(\mathbf{D}_2 \otimes \mathbf{I}) \right] \left( \text{vec}(\Phi^N) \right) \\ 187 \quad &:= \mathbf{K} \text{vec}(\Phi^N), \end{aligned}$$

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

191 (3) When  $d=3$ . Let  $\{i\}_{i=0}^N$ ,  $\{j\}_{j=0}^N$ , and  $\{k\}_{k=0}^N$  be the Gauss-Lobatto points of  $x_1$ ,  $x_2$ ,  
 192 and  $x_3$ , respectively. The collocation points are then  $(i, j, k)$ . In this case,  $\Phi^N$  is  
 193 approximated as

$$194 \quad (x_1, x_2, x_3) \approx \Phi^N(x_1, x_2, x_3) := \sum_{i=0}^N \sum_{j=0}^N \sum_{k=0}^N \Phi^N(i, j, k) P_i(x_1) P_j(x_2) P_k(x_3).$$

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

$$197 \quad \Phi^N(:, :, k) = \begin{bmatrix} \Phi^N(0, 0, k) & \Phi^N(0, 1, k) & \cdots & \Phi^N(0, N, k) \\ \Phi^N(1, 0, k) & \Phi^N(1, 1, k) & \cdots & \Phi^N(1, N, k) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi^N(N, 0, k) & \Phi^N(N, 1, k) & \cdots & \Phi^N(N, N, k) \end{bmatrix}.$$

199 With the n-mode multiplication in tensor algebra, we arrive at a compact representa-  
 200 tion of the approximation,

$$201 \quad \mathcal{K}\Phi^N = \mathbf{F}_1 \odot (\Phi^N \times_1 \mathbf{D}_1) + \mathbf{F}_2 \odot (\Phi^N \times_2 \mathbf{D}_2) + \mathbf{F}_3 \odot (\Phi^N \times_3 \mathbf{D}_3),$$

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

206 tensor representation as

$$\begin{aligned}
207 \quad \mathcal{K} \operatorname{vec}(\Phi^N) &= \operatorname{vec}(\mathbf{F}_1) \odot \left( (\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{D}_1) \operatorname{vec}(\Phi^N) \right) \\
208 \quad &+ \operatorname{vec}(\mathbf{F}_2) \odot \left( (\mathbf{I} \otimes \mathbf{D}_2 \otimes \mathbf{I}) \operatorname{vec}(\Phi^N) \right) \\
209 \quad &+ \operatorname{vec}(\mathbf{F}_3) \odot \left( (\mathbf{D}_3 \otimes \mathbf{I} \otimes \mathbf{I}) \operatorname{vec}(\Phi^N) \right) \\
210 \quad &= \left[ \operatorname{diag}(\operatorname{vec}(\mathbf{F}_1)) (\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{D}_1) \right. \\
211 \quad &+ \operatorname{diag}(\operatorname{vec}(\mathbf{F}_2)) (\mathbf{I} \otimes \mathbf{D}_2 \otimes \mathbf{I}) \\
212 \quad &+ \left. \operatorname{diag}(\operatorname{vec}(\mathbf{F}_3)) (\mathbf{D}_3 \otimes \mathbf{I} \otimes \mathbf{I}) \right] \left( \operatorname{vec}(\Phi^N) \right) \\
213 \quad &:= \mathbf{K} \operatorname{vec}(\Phi^N).
\end{aligned}$$

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

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

220 **3.2. Eigen-decomposition.** Now the eigenvalue problem of the Koopman operator in (2.5)  
221 is discretized as the eigenvalue problem of matrix  $\mathbf{K}$ , i.e.,  $\mathbf{K}\mathbf{v} = \tilde{\lambda}\mathbf{v}$ , where  $\tilde{\lambda} \in \mathbb{C}$  and  $\mathbf{v}$  is a  
222 complex vector. The vector  $\mathbf{v}$  is an approximation of  $\mathcal{K}$ 's eigenfunction evaluated at the  
223 collocation points and  $\tilde{\lambda}$  is the approximation of the associated eigenvalue of  $\mathcal{K}$ . The matrix  
224 form of the eigenvalue problem is

$$225 \quad (3.4) \quad \mathbf{K}\mathbf{V} = \mathbf{V}\Lambda,$$

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

234 **3.3. Constructing the solution.** Let us first consider  $d=1$ . By the eigen-decomposition,  
235 one can access values of eigenfunctions at the Gauss-Lobatto points  $\Xi := \{x_i\}_{i=0}^N$ , where  $x_0 <$   
236  $x_1 < \dots < x_N$ . Therefore,  $\psi(\mathbf{x}_0)$  can be approximated when  $x_0 \leq \mathbf{x}_0 \leq x_N$ . To avoid polynomial  
237 interpolation, ASK uses an even number for  $N$  and sets  $x_{N/2} = \mathbf{x}_0$ . Based on this setting, we  
238 consider a neighborhood of  $\mathbf{x}_0$  with radius  $r$ , i.e.,  $[x_0 - r, x_0 + r]$ , where  $r$  is tunable. Gauss-  
239 Lobatto points are then generated such that  $x_0 - r = x_0 < x_2 < \dots < x_{N/2} = \mathbf{x}_0 < \dots < x_N = x_0 + r$ .  
240 Thus,  $g_N$  is constructed as

$$241 \quad (3.5) \quad g_N(\mathbf{x}(t)) = \sum_{j=0}^N \tilde{c}_j \frac{1}{j} \binom{N}{j} (\mathbf{x}_0) e^{\tilde{\lambda}_j t} = \sum_{j=0}^N \tilde{c}_j \frac{1}{j} \binom{N}{N/2} e^{\tilde{\lambda}_j t} = \sum_{j=0}^N \tilde{c}_j (\mathbf{v}_j)_{N/2} e^{\tilde{\lambda}_j t},$$



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

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

$$245 \quad g(\mathbf{x}_0) \approx g_N(\mathbf{x}_0) = \sum_{j=0}^N \tilde{c}_j \mathbf{v}_j^N(\mathbf{x}_0),$$

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

$$247 \quad g(\mathbf{x}_i) \approx g_N(\mathbf{x}_i) = \sum_{j=0}^N \tilde{c}_j \mathbf{v}_j^N(\mathbf{x}_i), \quad i = 0, \dots, N,$$

248 where  $\mathbf{x}_i$  are the aforementioned Gauss-Lobatto points. Thus, we can obtain  $\tilde{c}_j$  by solving  
249 a linear system  $\mathbf{V}\mathbf{c} = \mathbf{g}(\mathbf{\Xi})$ , where  $\mathbf{V}$  is defined in (3.4),  $\mathbf{g}(\mathbf{\Xi}) = (g(\mathbf{x}_0), \dots, g(\mathbf{x}_N))^T$  and  
250  $\mathbf{c} = (\tilde{c}_0, \dots, \tilde{c}_N)^T$ . As an example, if  $g(\mathbf{x}) := \mathbf{x}$ , then  $\mathbf{g}(\mathbf{\Xi}) = (\mathbf{x}_0, \dots, \mathbf{x}_N)^T$ .

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

262 Next, since we vectorize matrix (or tensor)  $\Phi^N$  column by column (or slice by slice) as  
263 shown in subsection 3.1,  $\mathbf{v}_j(\mathbf{x}_0)$  is again approximated by the “middle” element of vector  
264  $\text{vec}(\Phi_j)$ , which leads to

$$265 \quad (3.6) \quad \mathbf{g}_N(\mathbf{x}(t)) = \begin{cases} \sum_{j=0}^{(N+1)^2-1} \mathbf{c}_j (\mathbf{v}_j)_{[(N+1)^2-1]/2} e^{-j t}, & d = 2; \\ \sum_{j=0}^{(N+1)^3-1} \mathbf{c}_j (\mathbf{v}_j)_{[(N+1)^3-1]/2} e^{-j t}, & d = 3. \end{cases}$$

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

275 **3.4. Adaptivity.** Since we apply a finite-dimensional approximation of the Koopman op-  
 276 erator and exploit the Lagrange interpolation to approximate the eigenfunctions, the accuracy  
 277 of the solution may decay as time evolves, especially for highly nonlinear systems. To further  
 278 improve the accuracy, we propose an adaptive approach to update  $\mathbf{V}$ ,  $\mathbf{\Lambda}$  and  $\mathbf{c}_j$ . The main idea  
 279 is to identify the time to repeat the procedure described in subsection 3.1– subsection 3.3. To  
 280 this end, we set check points  $0 < t_1 < t_2 < \dots < t_n < T$  to examine the “validity” of the neigh-  
 281 borhood of  $\mathbf{x}(t_k)$ . Specifically, the component of  $\mathbf{x}(t_k) = (x_1(t_k), \dots, x_d(t_k))^T$  is acceptable  
 282 if  $x_i(t_k) \in R_i$  where

$$283 \quad (3.7) \quad R_i := [L_i - r_i, U_i + r_i].$$

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

290 If  $x_i(t_k) \in R_i$  for all  $i$ , then  $R_1 \times \dots \times R_d$  is a valid neighborhood of  $\mathbf{x}(t_k)$ . Otherwise, we  
 291 update all  $L_i, U_i$  and reconstruct  $\tilde{\mathbf{N}}_j, \tilde{\mathbf{c}}_j$  to obtain  $\mathbf{x}(t)$  ( $t > t_k$ ) as follows:

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

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

300 The parameter  $\alpha$  decides how often we update the neighborhood and reconstruct the  
 301 solution. By construction, a larger  $\alpha$  demands updating the eigendecomposition more fre-  
 302 quently. The extreme case  $\alpha = 1$  enforces the update at every check point. In this work, we  
 303 set  $t_{k+1} - t_k \equiv \Delta t$ . Notably, since the solution is discretized *in space* instead of in time as  
 304 in conventional ODE solvers, the check points are different from time grids  $0 < t_1 < t_2 < \dots$   
 305 in those solvers. If we set  $k = 0$ , then no update is made, which indicates that the solu-  
 306 tion  $\mathbf{x}(t)$  only relies on the eigendecomposition based on  $\mathbf{x}_0$  (see the example pseudocode in  
 307 Appendix Appendix B).

308 **3.5. Properties of the algorithm.** In this work  $\mathbf{x}, \mathbf{f}, \mathbf{g}$  are real-valued functions. Now we  
 309 show that the solutions obtained by ASK are real numbers, although  $\mathbf{V}, \mathbf{\Lambda}, \mathbf{c}_j$  may contain  
 310 complex values. We start with reiterating a well-known conclusion:

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

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

318 Our main theorem is presented next:

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

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

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

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

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

341 **3.6. Algorithm summary.** As a summary, subsection 3.1 to subsection 3.3 present a  
 342 numerical scheme that solves an autonomous ODE system using the eigendecomposition and  
 343 a linear system solver. Subsection 3.4 introduces a heuristic adaptivity criterion to repeat the  
 344 aforementioned procedure at appropriate time points to further enhance the accuracy. We  
 345 conclude the algorithm in Algorithm 3.1.

346 In the ASK scheme, the neighborhood for all components must be updated in the adaptiv-  
 347 ity step. This is because we set the current state of each component to be the midpoint of the  
 348 corresponding neighborhood to avoid computing interpolation. Also, following the standard  
 349 practice in the spectral method, we generate Gauss-Lobatto points  $\xi_j$  and the associated dif-  
 350 ferentiation matrix  $\mathbf{D}_j$  on  $[-1, 1]$  first, and then scale them to  $[L_j, U_j]$  as  $\frac{U_j - L_j}{2}(\xi_j + 1) + L_j$  and  
 351  $\frac{2\mathbf{D}_j}{U_j - L_j}$  to improve the computational efficiency. Moreover, it is worth emphasizing again that  
 352 the isotropic setup (i.e., using the same number of Gauss-Lobatto points in each direction and  
 353 fix  $r_j \equiv r$ ) is not necessarily the optimal choice, and that the adaptivity in different directions  
 354 may improve the efficiency of the algorithm. This is beyond the scope of this work and will  
 355 be included in the future study.

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

**Algorithm 3.1** Adaptive spectral Koopman method**Require:**  $n, T, N, \mathbf{x}_0, r,$ 

- 1: Set check points at  $0 = t_0 < t_1 < \dots < t_n < T$ .
- 2: Let  $L_i = x_0^i - r_i, U_i = x_0^i + r_i$  and set neighborhood  $R_i$  as  $R_i = [L_i + r_i, U_i - r_i]$  for  $i = 1, 2, \dots, d$ , where  $r_i = r$ .
- 3: Generate Gauss-Lobatto points and differentiation matrix  $\mathbf{D}_i$  in  $[L_i, U_i]$  for  $i = 1, 2, \dots, d$ . Construct collocation points  $\Xi$  for  $d > 1$  using the tensor product rule. (For  $d = 1$ ,  $\Xi$  is the set of the Gauss-Lobatto points.)
- 4: Construct matrix  $\mathbf{K}$  using the formulas in subsection 3.1
- 5: Compute eigen-decomposition  $\mathbf{K}\mathbf{V} = \mathbf{V}\mathbf{\Lambda}$
- 6: Solve linear system  $\mathbf{V}\mathbf{C} = \mathbf{g}(\Xi)$ , where the  $l$ th column of matrix  $\mathbf{g}(\Xi)$  consists of the  $l$ th component of all collocation points (see subsection 3.3).
- 7: **for**  $k = 1, 2, 3, \dots, n$  **do**
- 8: Let  $x_j$  be the middle element of the  $j$ th column of  $\mathbf{V}$ . Construct solution at time  $t_k$  as  $\mathbf{x}(t_k) = \sum_j \mathbf{C}(j, :) x_j e^{\tilde{j}(t_k - t_{k-1})}$ , where  $\mathbf{C}(j, :)$  is the  $j$ th row of  $\mathbf{C}$ .
- 9: **if**  $(\mathbf{x}(t_k))_i \notin R_i$  for any  $i$  **then**
- 10:     Reset  $L_i = x_i(t_k) - r_i, U_i = x_i(t_k) + r_i$  and  $R_i = [L_i + r_i, U_i - r_i]$ .
- 11:     Repeat steps 3 – 6
- 12: **end if**
- 13: **end for**
- 14: **return**  $\mathbf{x}(T) = \sum_j \mathbf{C}(j, :) x_j e^{\tilde{j}(T - t_n)}$ .

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

368 **4.1. Cosine model.** The cosine model is a synthetic model invented for our demonstrative  
 369 purposes. The governing ODE is written as

$$370 \quad \frac{dx}{dt} = -0.5 \cos^2(x).$$

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

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

375 (b) test of  $n$ :  $N = 9, r = \frac{1}{20}$ ;

376 (c) test of  $r$ :  $n = 200, N = 9$ .

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

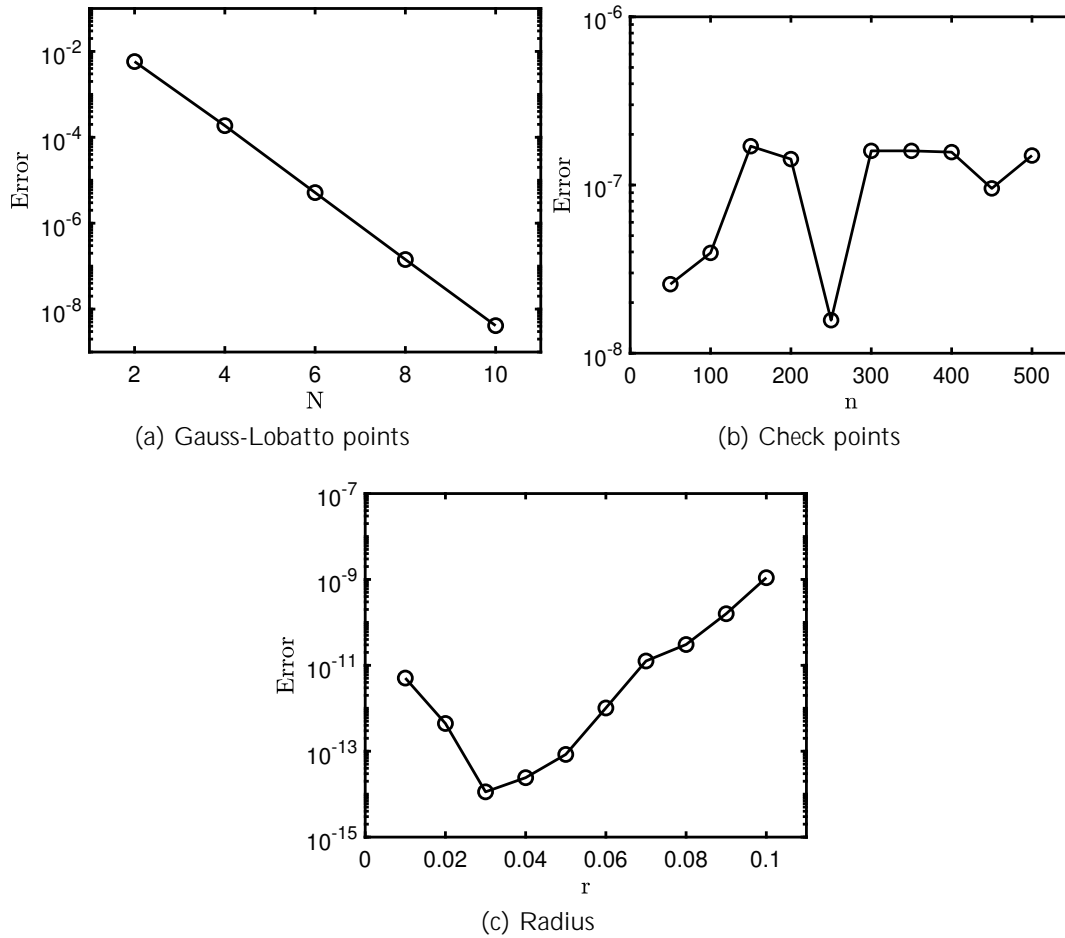


Figure 1: Cosine Model: (a) testing number of Gauss-Lobatto points  $N$ ; (b) testing number of check points  $n$ ; (c) testing radius  $r$ .

384 **4.2. Lotka-Volterra model.** The Lotka-Volterra equations model the interactive evolution  
385 of the population of prey and predators [2]. Specifically, it is defined by

$$\begin{aligned} 386 \quad & \frac{dx_1}{dt} = 1.1x_1 - 0.4x_1x_2, \\ 387 \quad & \frac{dx_2}{dt} = 0.1x_1x_2 - 0.4x_2. \end{aligned}$$

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

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

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

399 **4.3. Simple pendulum.** The simple pendulum is well studied in physics and mechanics.  
400 The movement of the pendulum is described by a second order ordinary differential equation,

$$\begin{aligned} 401 \quad & \frac{d^2}{dt^2} = -\frac{g}{L} \sin(\theta). \end{aligned}$$

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

$$\begin{aligned} 407 \quad & \frac{dx_1}{dt} = x_2, \\ 408 \quad & \frac{dx_2}{dt} = -\sin x_1. \end{aligned}$$

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

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

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

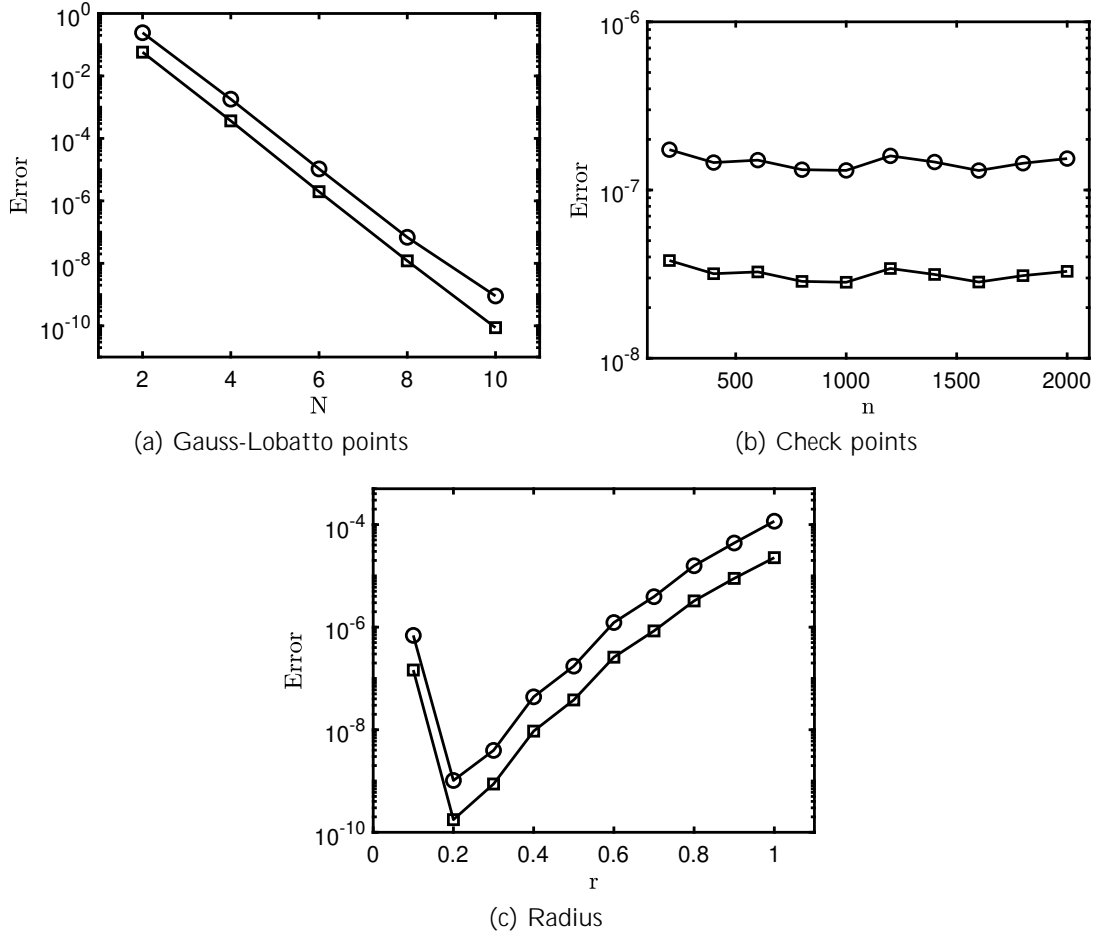


Figure 2: Lotka-Volterra Model: (a) testing number of Gauss-Lobatto points  $N$  (total number of collocation points is  $(N + 1)^2$ ); (b) testing number of check points  $n$ ; (c) testing radius  $r$ .  $\circ, \square$  denote  $x_1, x_2$ , respectively.

418 **4.4. Limit cycle.** The limit cycle is applied to model oscillatory systems in multiple re-  
 419 search fields [32]. Here, we follow the definition,

$$420 \quad \frac{dx_1}{dt} = -x_1 - x_2 + \frac{x_1}{\sqrt{x_1^2 + x_2^2}},$$

$$421 \quad \frac{dx_2}{dt} = x_1 - x_2 + \frac{x_2}{\sqrt{x_1^2 + x_2^2}}.$$

422

423 The closed-form solution is  $x_1(t) = \cos(t + \arcsin(x_1(0)))$  and  $x_2(t) = \sin(t + \arcsin(x_2(0)))$ .

424 We set  $\mathbf{x}(0) = (\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})^\top$  and  $T = 20$  in this example. The parameters in the experiments

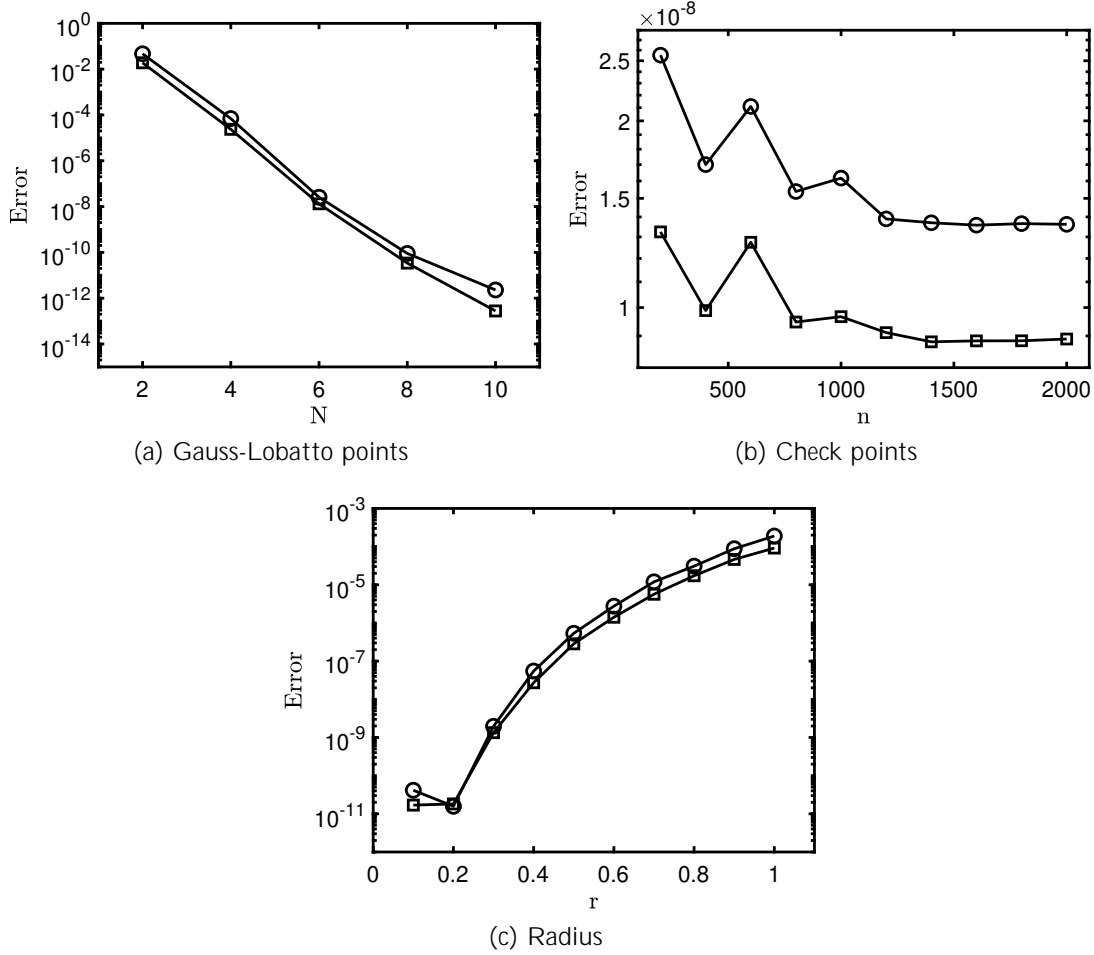


Figure 3: Simple Pendulum: (a) testing number of Gauss-Lobatto points  $N$  (i.e.,  $(N + 1)^2$  collocation points in total); (b) testing number of check points  $n$ ; (c) testing radius  $r$ .  $\circ, \square$  denote  $x_1, x_2$ , respectively.

425 are specified as follows:

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

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

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

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

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



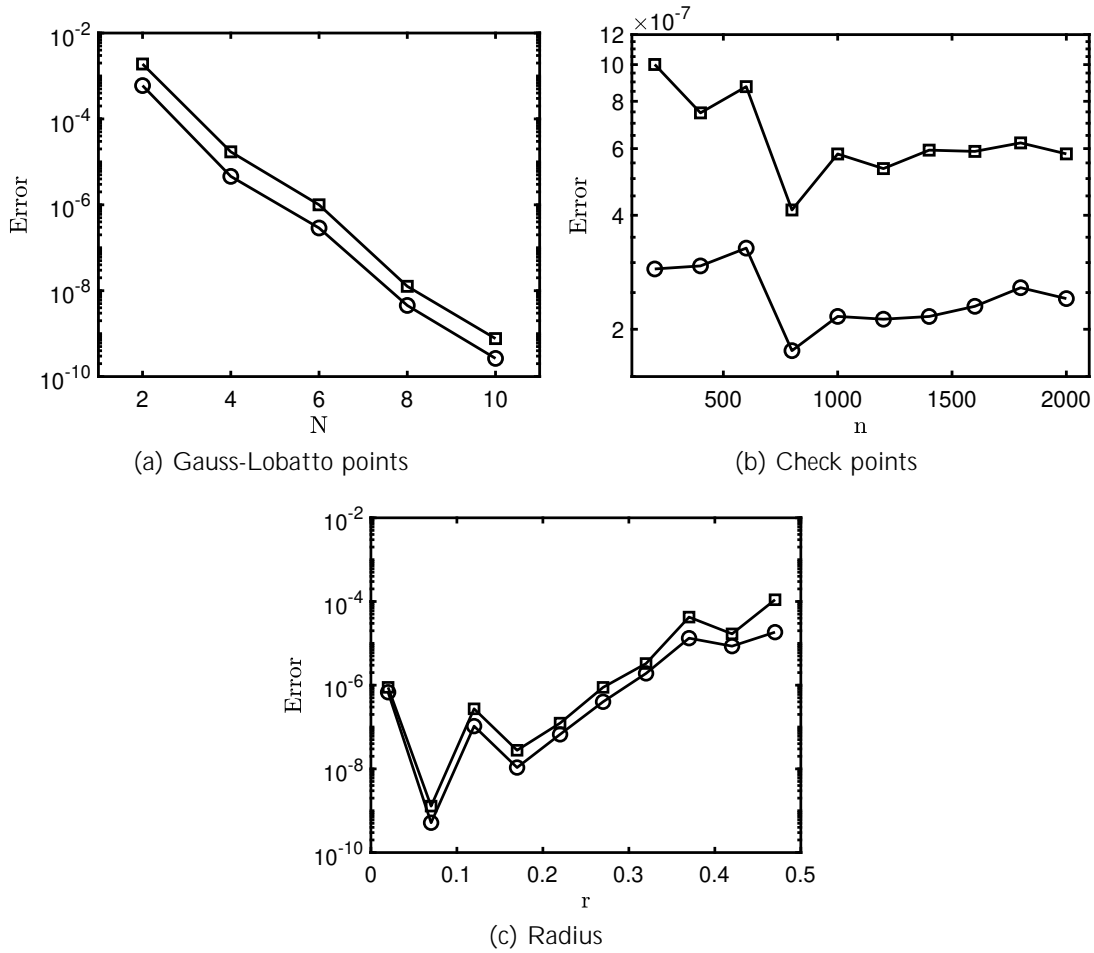


Figure 4: Limit Cycle: (a) testing number of Gauss-Lobatto points  $N$  (i.e.,  $(N+1)^2$  collocation points in total); (b) testing number of check points  $n$ ; (c) testing radius  $r$ .  $\circ, \square$  denote  $x_1, x_2$ , respectively.

433 Here, RK4 employed  $M = 200$  equidistant time points on  $[0, T]$ . As for ASK, we used  $N =$   
 434  $9, r = \frac{\sqrt{2}}{8}, \tau = 0.2$  and the check points are set to be the same as the time points in RK4.  
 435 With this set of parameters, ASK constantly outperforms RK4 significantly, as illustrated in  
 436 Figure 5. For both components  $x_1$  and  $x_2$ , the errors of ASK remain almost constant at the  
 437 level of  $10^{-10}$ . In comparison, the error of RK4 exhibited a periodic pattern, rising slowly  
 438 from  $10^{-6}$  to  $10^{-5}$ . Moreover, Figure 6 illustrates the evolution of the limit cycle model along  
 439 time. The path decided by the two components elevates spirally as time goes. If seen from  
 440 above, the cross section is an exact circle.

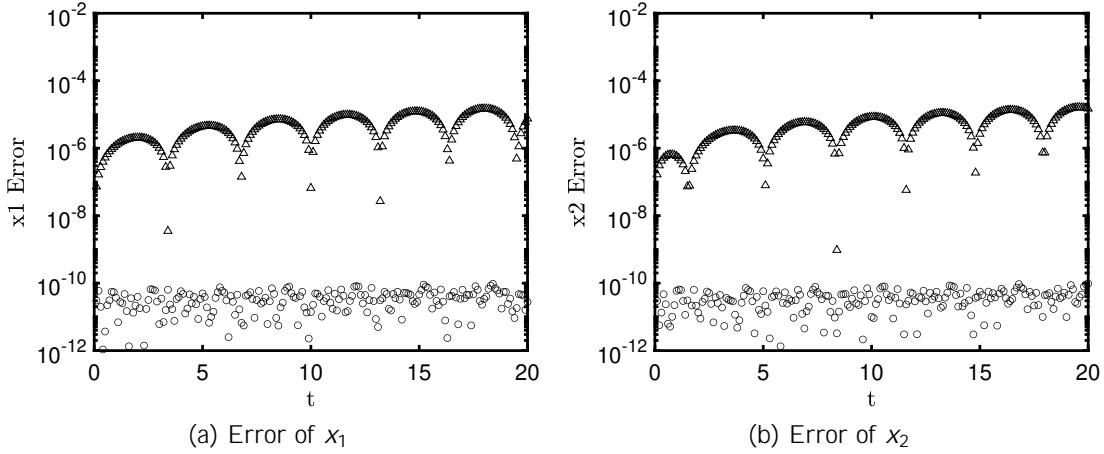


Figure 5: Errors of the limit cycle solution on  $[0, T]$ :  $\circ$  denotes ASK and  $\triangle$  denotes RK4.

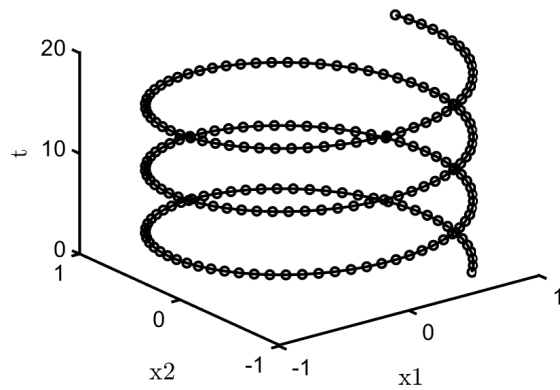


Figure 6: Limit cycle solution trajectory:  $\circ$  denotes ASK and  $-$  denotes the closed-form solutions.

441 **4.5. Kraichnan-Orszag model.** The Kraichnan-Orszag model comes from the problem  
 442 raised in [21]. This system is nonlinear and three-dimensional, defined by

443 
$$\frac{dx_1}{dt} = x_2 x_3,$$

444 
$$\frac{dx_2}{dt} = x_1 x_3,$$

445 
$$\frac{dx_3}{dt} = -2x_1 x_2.$$

446

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

449 (a) test of  $N$ :  $n = 400, r = 0.1$ ;

450 (b) test of  $n$ :  $N = 3, r = 0.1$ ;

451 (c) test of  $r$ :  $n = 400, N = 3$ .

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

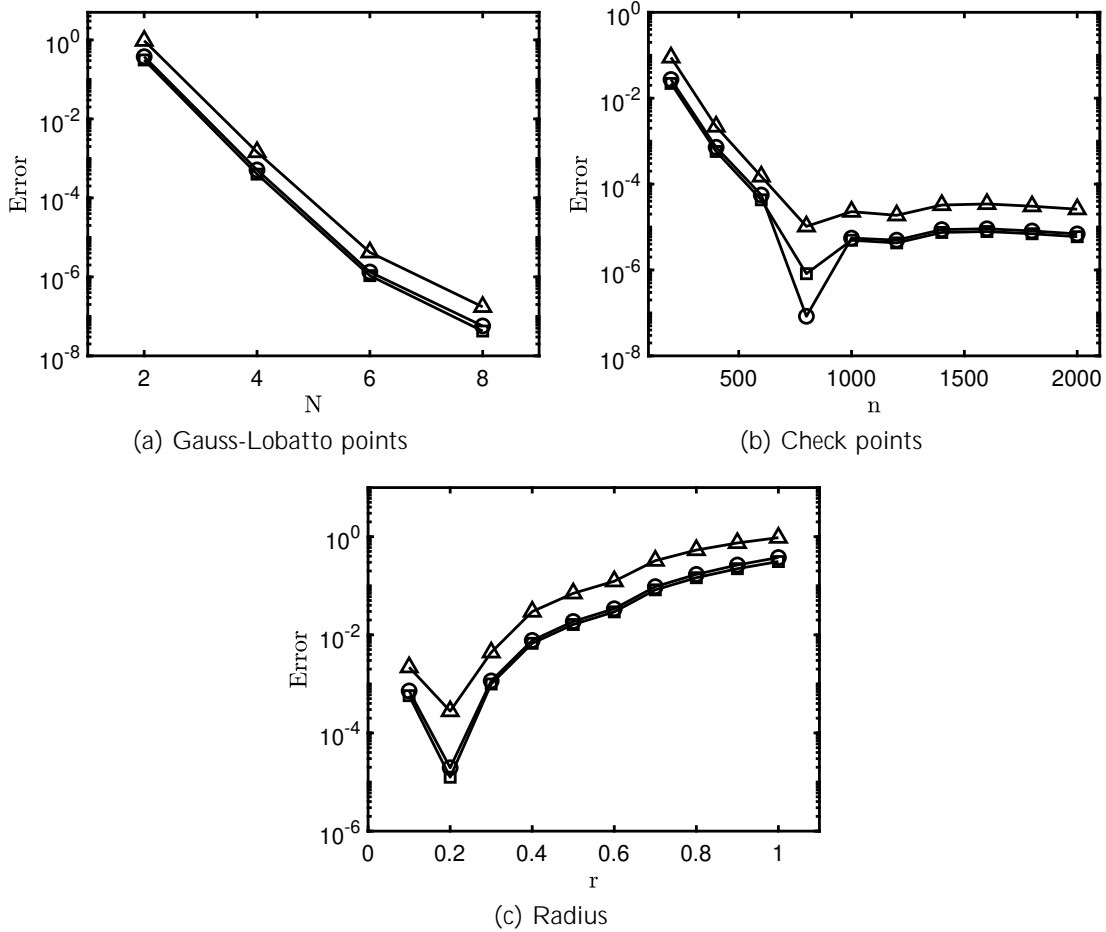


Figure 7: Kraichnan-Orszag model: (a) testing number of Gauss-Lobatto points  $N$  (i.e.,  $(N+1)^3$  collocation points in total); (b) testing number of check points  $n$ ; (c) testing radius  $r$ .  $\circ, \square, \triangle$  denote  $x_1, x_2, x_3$ , respectively

457 **4.6. Lorenz attractor.** The Lorenz attractor was first introduced by Lorenz [15]. It is a  
 458 highly chaotic system that models the turbulence in dynamic flows. The governing equations  
 459 are as follows,

$$\begin{aligned} 460 \quad & \frac{dx_1}{dt} = 10(x_2 - x_1), \\ 461 \quad & \frac{dx_2}{dt} = x_1(28 - x_3) - x_2, \\ 462 \quad & \frac{dx_3}{dt} = x_1x_2 - 3x_3. \end{aligned}$$

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

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

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

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

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

491 **4.7. Computational complexity.** By construction, the computational complexity of con-  
 492 ventional ODE solvers that approximate time integral is  $\mathcal{O}(M)$  where  $M$  is the number of  
 493 time steps. In other words, it is  $M$  multiplied by a constant that represents the number of  
 494 operations in each step and it varies according to the accuracy of the scheme. The computa-  
 495 tional complexity of ASK depends on the number of times that eigenfunctions are constructed  
 496 (and corresponding eigenvalues as well as coefficients are computed). In this construction pro-  
 497 cedure, ASK needs to perform the eigendecomposition and solve a linear system. For  $d = 1$

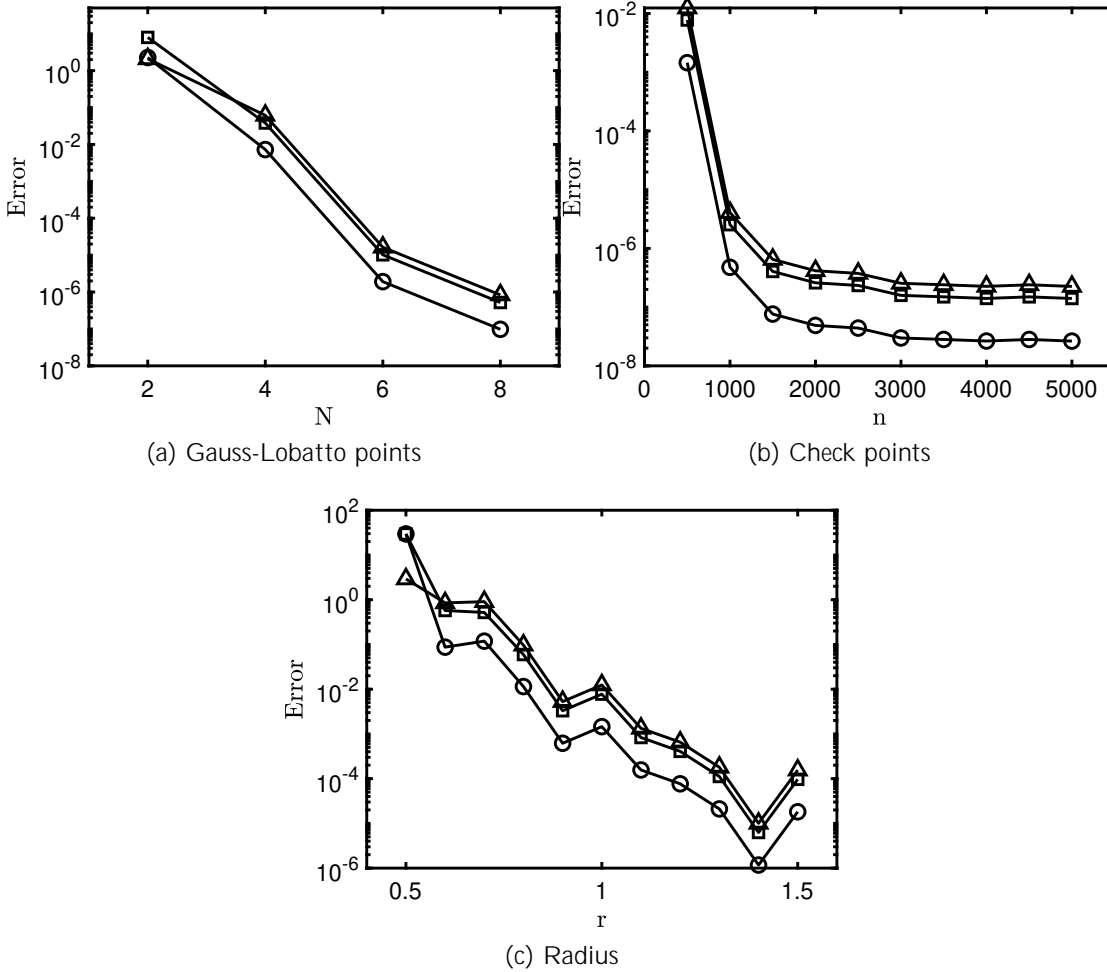


Figure 8: Loren attractor: (a) testing number of Gauss-Lobatto points  $N$  (i.e.,  $(N + 1)^3$  collocation points in total); (b) testing number of check points  $n$ ; (c) testing radius  $r$ .  $\circ, \square, \triangle$  denote  $x_1, x_2, x_3$ , respectively.

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

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

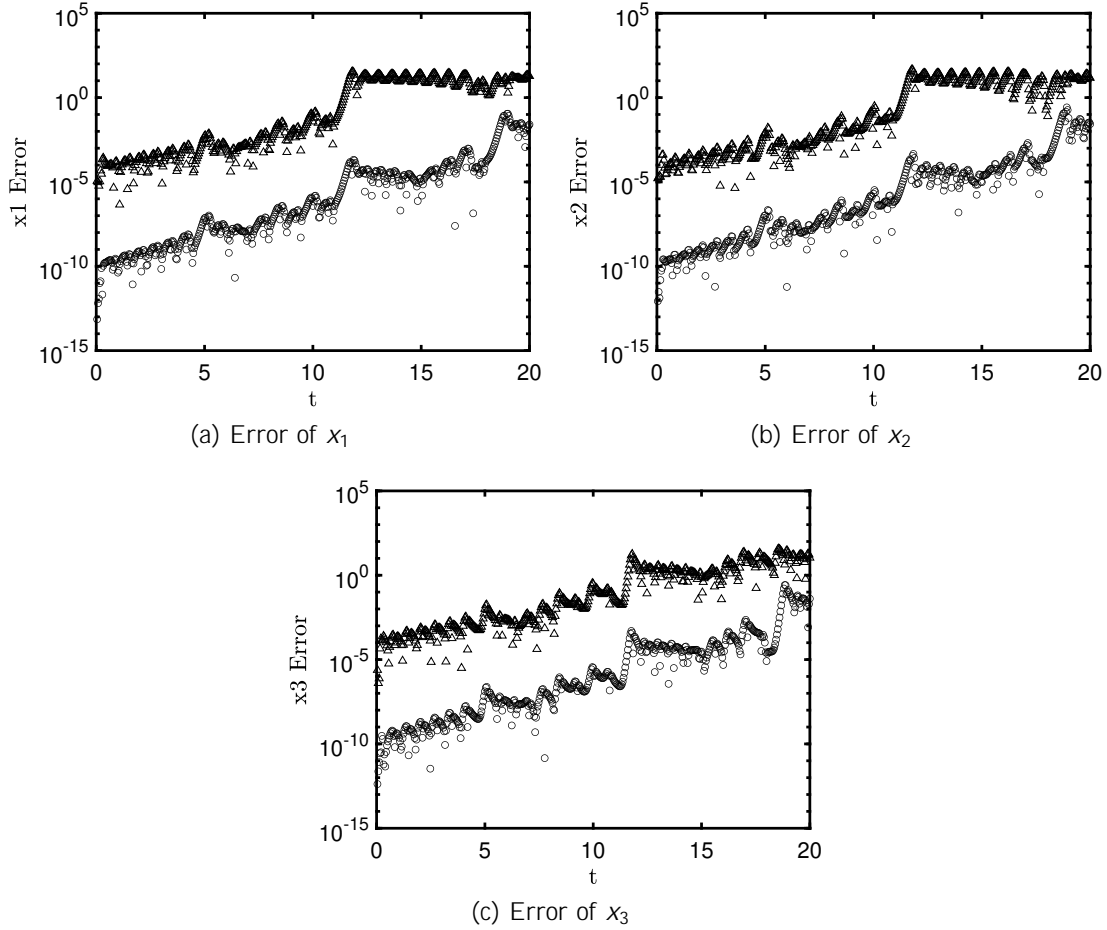


Figure 9: Error of Lorenz attractor solutions on  $[0, T]$ :  $\circ$  denotes ASK and  $\triangle$  denotes RK4.

506 against number of function calls. The cosine model ( $d=1$ ) and the simple pendulum ( $d=2$ )  
 507 exemplify the comparison and give the results in Figure 12. Here, the error in the simple  
 508 pendulum case was computed by  $\sqrt{\frac{e_1^2 + e_2^2}{2}}$ , where  $e_1, e_2$  are the errors in  $x_1, x_2$ , respectively.  
 509 The parameters in ASK are  $N = [4, 6, 8, 10], n = 50, r = \frac{1}{20}$  for the cosine model and  $N =$   
 510  $[2, 4, 6, 8], n = 60, r = \frac{\sqrt{2}}{12}$  for the simple pendulum. ASK is superior to the other two methods  
 511 in the cosine model. For the simple pendulum, RK9 is the most efficient method, and ASK  
 512 outperforms RK4 when number of function calls is beyond 2000.

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

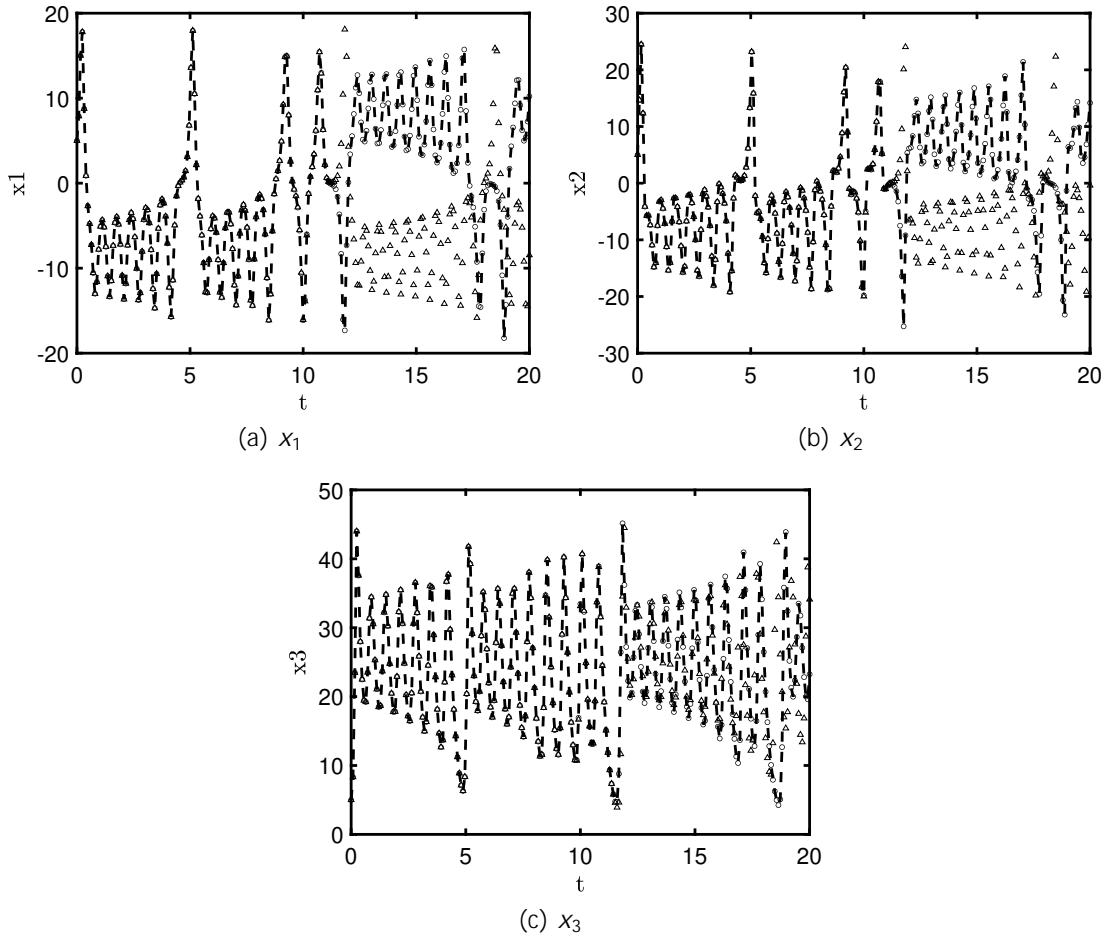


Figure 10: Lorenz Attractor Evolution:  $\circ$  and  $\triangle$  denote ASK and RK4, respectively;  $--$  denotes the reference solutions given by RK9

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

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

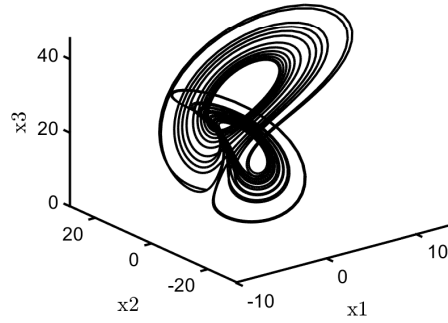


Figure 11: Lorenz Attractor 3D Visualization

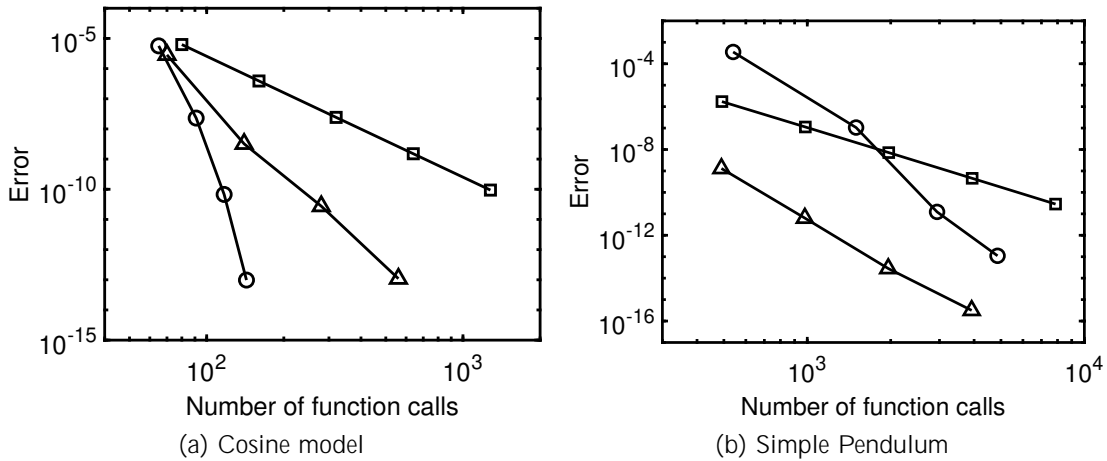


Figure 12: Test of Complexity: ○, □, and △ denote ASK, RK4, and RK9, respectively.

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

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



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

#### 544 **Appendix A. An example of the observable.**

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

$$\begin{aligned} 546 \quad \frac{dx_1}{dt} &= x_1, \\ 547 \quad \frac{dx_2}{dt} &= (x_2 - x_1^2). \end{aligned}$$

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

$$\frac{d\mathbf{y}}{dt} = \begin{bmatrix} 0 & 0 \\ 0 & -\alpha \\ 0 & 0 & 2\beta \end{bmatrix} \mathbf{y}.$$

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

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

557 Equivalently,

$$x_1 = e^{-\alpha t}, \quad x_2 = \frac{-2}{-2} e^{-\alpha t} + \frac{1}{-2} e^{2\beta t}.$$

560 **Appendix B. An example pseudocode.** We demonstrate a pseudo code (in MATLAB) of  
 561 solving  $\frac{dx}{dt} = \cos^2(x)$ , which summarizes the steps presented in [subsection 3.1–subsection 3.3](#).  
 562 The MATLAB code generating Chebyshev-Gauss-Lobatto points and the associated differentia-  
 563 tion matrix can be found in [29].

```

564 f = @(x) cos(x).^2; % Function f
565 x0 = pi/4; % Initial condition
566 r = 0.1; % Radius of the neighborhood (tunable)
567 N = 4; % Number of collocation points (N+1 in total)
568 T = 5; % Final time
569 % Generate collocation points and the differentiation matrix
570 % on [x0-r, x0+r]
571 [quad_pnt, diff_mat] = cheb(N, x0-r, x0+r);
572 % Compute eigenpairs of the Koopman operator

```

```

573     K = diag(f(quad_pnt))*diff_mat;
574     [eig_vec, eig_val] = eig(K, 'vector');
575     % Compute coefficients (Koopman modes)
576     coef = eig_vec\quad_pnt;
577     % Construct solutions at time T
578     sol = real(eig_vec(N/2,:) * coef * exp(eig_val * T));
579 When the adaptive update in ASK is activated (see subsection 3.4), we only need to repeat
580 this pseudocode (as a subroutine) with an updated initial condition  $x_0$  and final time  $T$ .

```

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

### 583 REFERENCES

- 584 [1] T. ASKHAM AND J. N. KUTZ, *Variable projection methods for an optimized dynamic mode decomposition*,  
585 SIAM Journal on Applied Dynamical Systems, 17 (2018), pp. 380–416.
- 586 [2] I. M. BOMZE, *Lotka-Volterra equation and replicator dynamics: a two-dimensional classification*, Biolog-  
587 ical cybernetics, 48 (1983), pp. 201–211.
- 588 [3] S. L. BRUNTON, B. W. BRUNTON, J. L. PROCTOR, AND J. N. KUTZ, *Koopman invariant subspaces*  
589 *and finite linear representations of nonlinear dynamical systems for control*, PloS one, 11 (2016),  
590 p. e0150171.
- 591 [4] S. L. BRUNTON, M. BUDIŠIĆ, E. KAISER, AND J. N. KUTZ, *Modern Koopman theory for dynamical*  
592 *systems*, arXiv preprint arXiv:2102.12086, (2021).
- 593 [5] M. BUDIŠIĆ AND I. MEZIĆ, *An approximate parametrization of the ergodic partition using time averaged*  
594 *observables*, in Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly  
595 with 2009 28th Chinese Control Conference, IEEE, 2009, pp. 3162–3168.
- 596 [6] M. BUDIŠIĆ, R. MOHR, AND I. MEZIĆ, *Applied Koopmanism*, Chaos: An Interdisciplinary Journal of  
597 Nonlinear Science, 22 (2012), p. 047510.
- 598 [7] J. C. BUTCHER AND G. WANNER, *Runge-Kutta methods: some historical notes*, Applied Numerical  
599 Mathematics, 22 (1996), pp. 113–151.
- 600 [8] B. FORNBERG, *A practical guide to pseudospectral methods*, no. 1, Cambridge university press, 1998.
- 601 [9] J. S. HESTHAVEN, S. GOTTLIEB, AND D. GOTTLIEB, *Spectral methods for time-dependent problems*,  
602 vol. 21, Cambridge University Press, 2007.
- 603 [10] G. KARNIADAKIS AND S. SHERWIN, *Spectral/hp element methods for computational fluid dynamics*, OUP  
604 Oxford, 2005.
- 605 [11] B. O. KOOPMAN, *Hamiltonian systems and transformation in Hilbert space*, Proceedings of the National  
606 Academy of Sciences of the United States of America, 17 (1931), p. 315.
- 607 [12] M. KORDA, M. PUTINAR, AND I. MEZIĆ, *Data-driven spectral analysis of the Koopman operator*, Applied  
608 and Computational Harmonic Analysis, 48 (2020), pp. 599–629.
- 609 [13] J. N. KUTZ, S. L. BRUNTON, B. W. BRUNTON, AND J. L. PROCTOR, *Dynamic mode decomposition:*  
610 *data-driven modeling of complex systems*, SIAM, 2016.
- 611 [14] J. N. KUTZ, X. FU, AND S. L. BRUNTON, *Multiresolution dynamic mode decomposition*, SIAM Journal  
612 on Applied Dynamical Systems, 15 (2016), pp. 713–735.
- 613 [15] E. N. LORENZ, *Deterministic nonperiodic flow*, Journal of atmospheric sciences, 20 (1963), pp. 130–141.
- 614 [16] B. LUSCH, J. N. KUTZ, AND S. L. BRUNTON, *Deep learning for universal linear embeddings of nonlinear*  
615 *dynamics*, Nature communications, 9 (2018), pp. 1–10.
- 616 [17] I. MEZIĆ, *Spectral properties of dynamical systems, model reduction and decompositions*, Nonlinear Dy-  
617 namics, 41 (2005), pp. 309–325.
- 618 [18] I. MEZIĆ, *Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space*  
619 *geometry*, Journal of Nonlinear Science, 30 (2020), pp. 2091–2145.
- 620 [19] H. NAKAO AND I. MEZIĆ, *Spectral analysis of the Koopman operator for partial differential equations*,

- 621 Chaos: An Interdisciplinary Journal of Nonlinear Science, 30 (2020), p. 113131.
- 622 [20] J. NATHAN KUTZ, J. L. PROCTOR, AND S. L. BRUNTON, *Applied Koopman theory for partial differential*  
623 *equations and data-driven modeling of spatio-temporal systems*, Complexity, 2018 (2018).
- 624 [21] S. A. ORSZAG AND L. BISSONNETTE, *Dynamical properties of truncated Wiener-Hermite expansions*, The  
625 Physics of Fluids, 10 (1967), pp. 2603–2613.
- 626 [22] J. PAGE AND R. R. KERSWELL, *Koopman analysis of Burgers equation*, Physical Review Fluids, 3 (2018),  
627 p. 071901.
- 628 [23] J. L. PROCTOR, S. L. BRUNTON, AND J. N. KUTZ, *Dynamic mode decomposition with control*, SIAM  
629 Journal on Applied Dynamical Systems, 15 (2016), pp. 142–161.
- 630 [24] C. W. ROWLEY, I. MEZIĆ, S. BAGHERI, P. SCHLATTER, AND D. S. HENNINGSON, *Spectral analysis of*  
631 *nonlinear flows*, Journal of fluid mechanics, 641 (2009), pp. 115–127.
- 632 [25] P. J. SCHMID, *Dynamic mode decomposition of numerical and experimental data*, Journal of fluid me-  
633 chanics, 656 (2010), pp. 5–28.
- 634 [26] J. SHEN AND T. TANG, *Spectral and high-order methods with applications*, Science Press Beijing, 2006.
- 635 [27] J. SHEN AND H. YU, *Efficient spectral sparse grid methods and applications to high-dimensional elliptic*  
636 *problems*, SIAM Journal on Scientific Computing, 32 (2010), pp. 3228–3250.
- 637 [28] J. SHEN AND H. YU, *Efficient spectral sparse grid methods and applications to high-dimensional elliptic*  
638 *equations ii. unbounded domains*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1141–  
639 A1164.
- 640 [29] L. N. TREFETHEN, *Spectral methods in MATLAB*, SIAM, 2000.
- 641 [30] J. H. TU, C. W. ROWLEY, D. M. LUCHTENBURG, S. L. BRUNTON, AND J. N. KUTZ, *On dynamic mode*  
642 *decomposition: Theory and applications*, Journal of Computational Dynamics, 1 (2014), pp. 391–421.
- 643 [31] J. H. VERNER, *Explicit Runge–Kutta methods with estimates of the local truncation error*, SIAM Journal  
644 on Numerical Analysis, 15 (1978), pp. 772–790.
- 645 [32] M. VIDYASAGAR, *Nonlinear systems analysis*, SIAM, 2002.
- 646 [33] M. O. WILLIAMS, M. S. HEMATI, S. T. DAWSON, I. G. KEVREKIDIS, AND C. W. ROWLEY, *Extending*  
647 *data-driven Koopman analysis to actuated systems*, IFAC-PapersOnLine, 49 (2016), pp. 704–709.
- 648 [34] M. O. WILLIAMS, I. G. KEVREKIDIS, AND C. W. ROWLEY, *A data-driven approximation of the Koopman*  
649 *operator: Extending dynamic mode decomposition*, Journal of Nonlinear Science, 25 (2015), pp. 1307–  
650 1346.
- 651 [35] D. WILSON AND J. MOEHLIS, *Isostable reduction with applications to time-dependent partial differential*  
652 *equations*, Physical Review E, 94 (2016), p. 012211.